P. 18774/92

Spis treści rocznika

12 1992

informatyka

Najlepsze życzenia świąteczne i noworoczne wszystkim Czytelnikom i Autorom składa

Zespół redakcyjny

WYDAWNICTWO SIGMA NOT

Cena 18 000 zł

informatyka

nr 12 1992
Grudzień
Rok wydania XXVII

Miesięcznik
ISSN 0542-9951
INDEKS 36124

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET
mgr inż. Piotr FUGLEWICZ
mgr Teresa JABŁOŃSKA
(sekretarz redakcji)
Władysław KLEPACZ
(redaktor naczelny)
dr inż. Wojciech MOKRZYCKI
mgr inż. Jan RYŻKO
dr Zdzisław SZYJEWSKI
mgr Hanna WŁODARSKA – MARCZENKO

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu 00-950 Warszawa ul. Mazowiecka 12 telefon: 27-43-66 telefaks: 19-21-87 teleks: 814877

W numerze:

	Strona
Tam gdzie minus oznacza dzielenie – *Bogusław Jackowski, Marek Pyćko*	1
Uniwersalny monitor szyny dla dydaktycznego systemu mikroprocesowego – *Artur Chojnacki, Zbigniew Kalisiak*	4
NFS – wspólna platforma systemów UNIX i DOS – *Grzegorz Kowalczyk*	10
Współpraca pakietu PC NFS z siecią Novell – *Mirosław Szczypiński*	13
SYSTEC'92 – *Mariusz Postół*	14
Modelowanie wiedzy w metodyce KADS – *Jerzy Surma*	15
Operacje wyszukiwania w bazie danych systemu GFG – *Zdzisław Wrzeszcz*	18
Diagnostyka systemów informacyjnych (2). Model diagnostyki systemu informacyjnego – *Agnieszka Szewczyk*	22
Roczny spis treści (wewnątrz numeru)	I–IV

W najbliższych numerach:

- **Dariusz Caban opisuje środki łączności w przemysłowych rozproszonych systemach sterowania.**
- **Iwona Dąbrowska i Dorota Depowska zajmują się podstawowymi problemami rozpoznawania pisma drukowanego w systemach off-line.**
- **Bohdan Macukow i Tomasz Berus prezentują kilka zaawansowanych modeli sieci neuronowych wykorzystywanych dla celów komercyjnych.**
- **Waldemar Wieczerzycki charakteryzuje obiektowe bazy danych.**

Warunki prenumeraty

Przyjęcie prenumeraty – wyłącznie na podstawie dokonanej wpłaty na drukach dostarczanych dotychczasowym prenumeratorom przez Wydawnictwo, lub nowym – po uprzednim zgłoszeniu zapotrzebowania (pisemnie lub telefonicznie) w Zakładzie Kolportażu Wydawnictwa.

Blankiet wpłaty – powinien zawierać następujące informacje: dokładna nazwa i adres (z kodem pocztowym) zamawiającego, tytuły zamawianych czasopism, ich liczbę i okres prenumeraty.

Wpłata – zgodnie z podanymi cenami należy dokonać w banku lub w UPT na konto podane na naszym blankiecie, tj:
Państwowy Bank Kredytowy III O/Warszawa nr: 370015-1573-139-11

Prenumeratorzy zbiorowi – osoby prawne obowiązują blankiety „Wpłata-Zamówienie". Cena normalna.

Prenumeratorzy indywidualni – osoby fizyczne obowiązują blankiety typu przekazy dla wpłat na rachunki bankowe. Cena normalna.

Prenumerata ulgowa – zgodnie z podaną ceną ulgową przysługuje wyłącznie osobom fizycznym, będącym członkami SNT, studentom i uczniom szkół zawodowych. Uczniowie szkół ogólnokształcących mogą zamówić w prenumeracie ulgowej tylko miesięcznik „Aura"

Uwaga! w podanym okresie prenumeraty można zamówić tylko po jednym egzemplarzu z każdego tytułu.

Prenumerata ze zleceniem wysyłki za granicę – cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa od ceny normalnej.

Należy podać dokładny adres odbiorcy za granicą.

Terminy przyjmowania prenumeraty:
– do 10 listopada na I, II, III, IV kwartał następnego roku
– do 28 lutego na II, III, IV kwartał br.
– do 31 maja na III i IV kwartał br.
– do 31 sierpnia na IV kwartał br.

Zmiany w prenumeracie, np. zmiana liczby tytułów, liczby egzemplarzy, rezygnacja z prenumeraty, można zgłaszać tylko w podanych terminach z mocą obowiązującą od następnego kwartału.

Egzemplarze archiwalne (z lat ubiegłych)
Można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa, ul. Mazowiecka 12 (tel. 26-80-16) lub zamówić pisemnie w Zakładzie Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek lub za zaliczeniem pocztowym.

Informacji o prenumeracie udziela: Zakład Kolportażu Wydawnictwa SIGMA-NOT Spółka z o.o., 00-716 Warszawa, ul. Bartycka 20, skr. 1004. Telefony: 40-00-21 wewn. 293, 295, 299 lub 40-30-86, 40-35-89.

Wstępna cena jednego egzemplarza na 1992 rok: normalna – 18 000 zł, ulgowa – 13 500zł

Wartość prenumeraty (w zł):
Normalna: kwartalna – 54 000, półroczna 108 000, roczna 216 000
Ulgowa: kwartalna – 40 500, półroczna 81 000, roczna 162 000

Uwaga: W przypadku zmiany cen w okresie objętym prenumeratą, prenumeratorzy zobowiązani są do dopłaty różnicy cen.

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o.

P. 1877/92

BOGUSŁAW JACKOWSKI
„Polski TEX"
Sopot
MAREK PYĆKO
Wydawnictwo „DO"
Warszawa

Tam gdzie minus oznacza dzielenie

Ten nieco przewrotny tytuł dotyczy oczywiście przenoszenia, czyli inaczej dzielenia wyrazów, zwyczajowo zaznaczanego w tekstach kreską przypominającą minus. Automatyczne dzielenie wyrazów należy do podstawowych zadań wykonywanych przez programy komputerowe służące do składania tekstów. Programy takie, niezbyt ściśle określane jako programy DTP (ang. *Desktop Publishing*), zdobywają ostatnio w Polsce sporą popularność. Jednym z najciekawszych algorytmów dzielenia wyrazów, a przy tym dokładnie opisanych i udokumentowanych, posługuje się system TEX (autorstwa D.E. Knutha z Uniwersytetu Stanforda w USA). Jego „ciekawość" polega na tym, że jest to algorytm uniwersalny, tzn. dający się przystosować do dzielenia wyrazów w różnych językach, a jednocześnie zdumiewająco prosty i efektywny w implementacji. Warto się temu algorytmowi przyjrzeć.

Reguły dzielenia wyrazów w języku polskim

Żeby zrozumieć, jak trudne zadanie postawił przed sobą Knuth, zajmijmy się przez chwilę regułami dzielenia wyrazów w języku polskim. M. Szymczak („Słownik ortograficzny języka polskiego", PWN, Warszawa, 1986) pisze:

(...) *Dzielenie wyrazów przy przenoszeniu wyrazów z jednego wiersza do następnego opiera się w ortografii polskiej na dwóch kryteriach: fonetycznym i morfologicznym. Kryterium fonetyczne nakazuje przenoszenie części wyrazu z jednego wiersza do drugiego zgodnie z podzielnością na sylaby, np. bu-rza, nie-wy--trzy-ma-nie, li-tość, po-na-pi-sy-wa-li-by, za-nie-po-ko-jo-ny, wy-so-ko, le-piej, te-go, prze-ciw. Kryterium morfologiczne nakazuje przenoszenie części wyrazu z jednego wiersza do drugiego zgodnie z podzielnością na przedrostek i rdzeń, a w wyrazach złożonych – w miejscu złożenia, np. przed-murze, pod-punkt, wy-brze-że, roz-łąka, od-dźwięk, przy-kład, za-płata, na-trafić, wy-ścigi, o-brona, bez-wstyd, u-kładowy, po-dróż, pod-oficer, pod-oddział, na-przód, pod-o-pieczny, po-dmuch, noc-leg, konio--krad,* (...) przy czym (...) *kryterium morfologiczne jest nadrzędne w stosunku do kryterium fonetycznego* (...)

Zasady ogólne wyglądają prosto i przejrzyście. Omówienie zasad szczegółowych zajmuje Szymczakowi pięć stron i o ile człowiekowi mogłoby to wystarczyć, to jednak zalecenia, takie jak:

(...) *Jeżeli przejrzystość podziału na przedrostek i rdzeń jest w dzisiejszej świadomości językowej zatarta* (podkr. aut.), *grupy spółgłoskowe znajdujące się na granicy cząstek dzielimy dowolnie, np. o-błok a. ob-łok, o-tręby a. ot-ręby,* (...)
bardzo trudno zalgorytmizować.

Spróbujmy zastanowić się czy nie ma takich zasad, od których nie byłoby wyjątków. Dobrym kandydatem wydawać by się mogła reguła niedzielenia głosek *cz*, *dz*, *rz* oraz *sz*. Niestety, są wyjątki od tej reguły: *tysiąc-złotowy, pod-zwrotnikowy, mar-znąć* i *em-es-zet*. Podobnie miękkie *ni* może ulec rozbiciu, na przykład w słowie *in-iekcja*.

To może chociaż da się ustalić, że sylaba nie może składać się z samych spółgłosek? Jest to w zasadzie dobra reguła, ale ma (co najmniej) jeden wyjątek: szkocki przedrostek *Mc* wolno oddzielić. Na przykład, nazwisko *McMillan* wolno podzielić *Mc-Millan*. Oczywiście implikuje to kłopoty z rzymskim zapisem liczb – rok MCMXCII komputer mógłby uznać za szkockie nazwisko.

W takim razie może istnieją nie zakazy, a nakazy dzielenia, od których nie ma wyjątków? Np. nakaz rozdzielania par identycznych spółgłosek (*świec-cy, Jagiel-ło, łam-my, pan-na, ar-ras, las-so, get-to* itp.). Odpowiedź, jak łatwo było przewidzieć, brzmi: „nie". Np. kryterium morfologiczne nakazuje podział *ode-ssie*, a nie *odes-sie*, chyba że chodzi o *Odessę*. A jeśli się zgodzić, że przedrostek rodzimy musi być oddzielony, to będzie kłopot ze słowem *zeppelin*, bo jak komputer miałby się domyślić, że w tym przypadku nie chodzi o przedrostek *ze*? Mógłby wprawdzie wiedzieć, że żadne słowo w języku polskim nie zaczyna się od *pp* (chociaż jest słowo zaczynające się od trzech spółgłosek, z czego dwie pierwsze to *ww* – proszę zgadnąć jakie to słowo). Ale jeśli jakiś autor chciałby oddać mowę jąkającego się bohatera pisząc np. *zeppsuuty*, to co wtedy?

Niejednoznaczności typu *odessie* stanowią dla komputera trudności praktycznie nie do pokonania. Choć jest ich niewiele, to jednak zdarzają się i utrudniają życie, np. *podróżować* – być w podróży lub pokryć różem lica; *odziewać* – ubierać lub odpowiedzieć w taki sam sposób komuś, kto ziewnął; *podrobić* – np. karmę ptactwu lub podpis; *Tarzanie* – słowo to może oznaczać wzywanie znanego bohatera komiksów lub czynność tarzania się; *narwali* – to słowo z kolei można rozumieć jako czas przeszły dokonany albo dopełniacz liczby mnogiej.

W sytuacjach tego typu jedynie kontekst semantyczny decyduje o podziale słowa. Rozsądnie jest nie żądać zbyt wiele od programów składających teksty i zostawić człowiekowi możliwość ingerencji w bardziej skomplikowanych przypadkach.

Sporo trudności nastręcza dzielenie nazwisk i nazw własnych. O nazwiskach szkockich już była mowa. Ale i polskie nazwy potrafią sprawić kłopot. Na przykład, czy w nazwisku *Utnik* głoskę *u* należy traktować jako przedrostek, tak jak w słowie

u-tnie? Szczególnie trudne do podziału są słowa będące zbitką dwóch lub więcej słów. Zalecany jest oczywiście podział w miejscu złożenia: *noc-leg, trzech-set-letni* itp. Zdarzają się jednak czasem bardzo osobliwe nazwy własne, np. *Wierzch-las*. Chociaż taki właśnie podział wydaje się uzasadniony, to przecież trudno dać głowę, że nazwy tej nie da się wywieść od wierzenia i chlastania. Podobnie gdyby jakaś miejscowość nazywała się *Szynkwas*, to nazwa mogłaby się wywodzić ostatecznie – choć jest to mało prawdopodobne – od *szyn* i *kwasu*.

Zeppelin i *szynkwas* to przykłady słów obcych bądź obcego pochodzenia, które zadomowiły się w naszym języku. Słowa tej kategorii, takie jak *er-zac, jazz-man* czy *soft-ware*, to źródło nowych utrapień dla kogoś, kto chciałby sformalizować reguły dzielenia wyrazów w języku polskim.

W tej sytuacji nie dziwią wyniki testów porównawczych W. Puzy (W. Puza, praca magisterska pt. „Analiza porównawcza wybranych programów Desktop Publishing (DTP)", Instytut Poligrafii Politechniki Warszawskiej), który stwierdził, co następuje:

(...) *Test sprawdzający poprawność dzielenia polskich słów wyodrębnił 3 różne poziomy skuteczności testowanych programów*:
– *Cyfroset*, TEX – ok. 10% błędów,
– *Dywiz (dedykowany dla Quark'a)* – powyżej 20% błędów,
– *Bizon (dedykowany Calamusowi), VP Commander (dedykowany Venturze) – powyżej 50% błędów*. (...)

Tak więc pozornie trywialny problem okazuje się w praktyce trudnym orzechem do zgryzienia.

Można oczywiście polemizować z wytycznymi językoznawców. Zalecany przez Szymczaka podział *za-jrzeć* lub *po-jmać* wydaje się dziś anachroniczny. Pójście dalej tym tropem, przy równoczesnym zasłanianiu się „wymaganiami komputerów", powoduje, że zbyt kuszące staje się maksymalne upraszczanie reguł, oczywiście z uszczerbkiem dla języka polskiego.

Autor TEX-a obrał zupełnie inną drogę: szukał takiego algorytmu, który byłby w zasadzie niezależny od reguł dzielenia wyrazów; reguły dzielenia miałyby być określone za pomocą danych do tego algorytmu. Algorytm spełniający te wymagania skonstruował F. M. Liang (rozprawa doktorska pt. „Word Hy-phen-a-tion by Com-put-er", Uniwersytet Stanforda, USA, 1983). Udało mu się nawet coś więcej: dane są stosunkowo łatwe do utworzenia, nie jest do tego bynajmniej potrzebna wiedza informatyczna.

Jak TEX dzieli wyrazy

Algorytm dzielenia wyrazów jest niezwykle prosty. Oczywiście przy założeniu, że odpowiednie dane zostały utworzone. Załóżmy więc, że te dane mamy. Składa się na nie zbiór ciągów utworzonych na przemian z liter i cyfr, np.: 2*s*0*z*1*z*0, 2*s*0*z*0*l*0*n*0, 2*t*1*ć*0 itp. Zamiast cyfry skrajnej może wystąpić kropka, np.: *w*0*e*3*s*2., *w*0*w*8, 8*r*0*s*0*z*., 8*r*0*z*0*ł*. itp. Ciągi te nazywać będziemy wzorcami.

Przypuśćmy teraz, że mamy dane jakieś słowo, np. *odkaszlnąć*. Najpierw zamieniamy je na postać taką, jaką mają wzorce, wstawiając pomiędzy literami cyfrę 0, a na brzegach kropki, otrzymując .*o*0*d*0*k*0*a*0*s*0*z*0*l*0*n*0*ą*0*ć*. (kropka – jak stąd widać – oznacza skraj słowa). Następnie wyszukujemy w naszym zbiorze danych wszystkie wzorce, które na to słowo dają się nałożyć w taki sposób, że zgadzają się położenia liter i kropek, ale niekoniecznie cyfr; w szczególności cyfra może się nakładać na kropkę, ale nie na literę. Powiedzmy, że znaleźliśmy następujące wzorce: .*o*2*d*2, .*o*0*d*3*k*2, 0*ą*1, 2*d*1*k*0, 2*l*1*n*0, 2*s*0*z*1*l*0, 2*s*0*z*0*l*0*n*0, 8*ć*., 0*a*1, 0*o*1, 0*s*4*z*0. Jeśli je nałożymy na słowo .*o*0*d*0*k*0*a*0*s*0*z*0*l*0*n*0*ą*0*ć*. – i spośród cyfr nakładających się na siebie weźmiemy największe, to otrzymamy .*o*2*d*3*k*2*a*2*s*4*z*2*l*1*n*0*ą*8*ć*. – i to już wszystko. Zgodnie z algorytmem Lianga słowo wolno podzielić jedynie w takim miejscu, w którym pojawia się cyfra nieparzysta. Oznacza to, że algorytm dopuszcza w tym wypadku dwa punkty podziału: *od-kaszl-nąć*.

Wzorce przechowywane są w pamięci komputera jako struktura drzewiasta (dokładniej *trie*; p. D. E. Knuth, „The Art of Computer Programming", tom 3, „Sorting and Searching"). Pozwala to na znaczne upakowanie danych przy zachowaniu szybkiego dostępu do informacji, a równocześnie struktura danych i algorytm są stosunkowo łatwo implementowalne.

Warto podkreślić, że dzięki efektywności zastosowanej metody TEX może przechowywać w pamięci kilka różnych zestawów wzorców i dzielić wyrazy w kilku językach w obrębie jednego dokumentu (a nawet akapitu).

Problem jedynie w tym, skąd wziąć wzorce?

Automatyczne tworzenie danych dla algorytmu dzielenia wyrazów

Główna część pracy Lianga dotyczyła metody tworzenia wzorców na podstawie słownika zawierającego słowa poprawnie podzielone. Przygotowanie takiego słownika jest oczywiście czaso- i pracochłonne, ale za to dalsza część zadania jest już względnie łatwa. W największym uproszczeniu algorytm Lianga można zapisać następująco:

```
m:= maksymalna długość wzorca
for c := 1 to 9 do
begin
  ustal kryteria akceptowalności dla wzorców
  for l := 1 to m do
  begin

    dla każdego słowa w słowniku sprawdź, czy nie dostarcza
    ono informacji o możliwości wstawienia cyfry c we wzor-
    cach długości l przy zadanych kryteriach akceptowalności

  end
end
```

Innymi słowy program uzupełnia zestaw wzorców kolejno dla coraz większych cyfr i coraz dłuższych wzorców, zostawiając użytkownikowi na każdym etapie możliwość ingerencji. Kryteria akceptowalności określa się przez podanie trzech liczb: wagi dla podziałów poprawnych w_p, wagi dla podziałów niepoprawnych w_n i poziomu akceptowalności p. Wzorzec jest akceptowany, gdy

$$k_p w_p - k_n w_n \geqslant p,$$

gdzie k_p i k_n oznaczają odpowiednio liczbę poprawnych i niepoprawnych podziałów generowanych przez dany wzorzec.

W niektórych przypadkach może się zdarzyć, że program nie znajdzie zestawu wzorców dzielącego poprawnie wszystkie słowa. Powodem mogą być błędne dane (np. w słowniku może się znaleźć dwa razy to samo słowo, ale inaczej podzielone) bądź niewłaściwy dobór liczb w_p, w_n oraz p w kolejnych iteracjach. Różne strategie dobierania wartości tych współczynników wpływają na objętość wynikowego zestawu wzorców. Minimalizacja liczby wzorców w wygenerowanym automatycznie zestawie wymaga pewnej wprawy i – oczywiście – przestu-

diowania pracy doktorskiej Lianga, gdzie zagadnienie to jest szczegółowo omówione.

Jak widać „nauczenie" TEX-a zasad dzielenia wyrazów w danym języku jest właściwie kwestią odpowiednio zasobnego słownika podzielonych poprawnie wyrazów. Dostarczenie takiego słownika to zadanie dla językoznawców. Natomiast dalszy etap jest ideowo prosty, nie wymaga znajomości technik programowania ani żadnych innych specjalnych umiejętności. Potrzebna jest jedynie cierpliwość.

Ręczne tworzenie danych dla algorytmu dzielenia wyrazów

Możliwość automatycznego tworzenia zestawu wzorców nie wyklucza możliwości stworzenia takiego zestawu ręcznie. Początkowo wydawało się, że język polski ma na tyle regularne zasady dzielenia wyrazów, że łatwiej będzie utworzyć zestaw wzorców w ten właśnie sposób.

W 1988 r. J. Désarménienowi udało się zamknąć reguły dzielenia wyrazów dla języka francuskiego w zestawie liczącym 804 wzorce i dla języka włoskiego w zestawie liczącym zaledwie 88 wzorców, podczas gdy standardowy zestaw wzorców dla amerykańskiego TEX-a zawiera ich 4447 (J. Désarménien, „The use of TEX in French: hyphenation and typography", w: D. Lucarella (ed.), „TEX for Scientific Documentation, Proc, of the First European Conference, Addison-Wesley, 1984). Sukces podejścia Désarméniena zainspirował H. Kołodziejską, która w 1987 r. opublikowała listę 2168 wzorców dla języka polskiego (H. Kołodziejska, „Dzielenie wyrazów w systemie TEX", Sprawozdania Instytutu Informatyki Uniwersytetu Warszawskiego, nr 165, 1987).

Wynik Kołodziejskiej zdawał się potwierdzać wstępne założenia. Niestety. Po kilku latach intensywnego testowania (maczali w tym palce m.in. autorzy niniejszej pracy) wzorców znacznie przybyło. Po gruntownej przeróbce liczebność zestawu wzorców wzrosła do 4053 i wolno przypuszczać, że to jeszcze nie koniec.

W tych warunkach zasadne staje się pytanie, czy nie należałoby zrewidować założenia o regularności zasad podziałów słów w języku polskim. Jest prawdopodobne, że program Lianga mógłby wygenerować mniej liczny zestaw wzorców.

Niemniej, jak wynikało z porównań poczynionych przez Puzę (p. wyżej), zestaw wzorców dla języka polskiego w swojej obecnej postaci daje wyniki zadowalające. Zważywszy, że Puza w istocie dysponował jedną z wersji pośrednich zestawu i że zestaw aktualny jest znacznie udoskonalony w stosunku do tamtej wersji, można oczekiwać jeszcze lepszych rezultatów.

Najaktualniejsza wersja tego zestawu wchodzi w skład pakietu MEX (polskiej adaptacji TEX-a). Pakiet ten jest dystrybuowany w postaci źródłowej jako produkt public domain przez Grupę Użytkowników Systemu TEX (GUST).

Inne aspekty automatycznego dzielenia wyrazów

Automatyczne znajdowanie miejsc podziału słów nie wyczerpuje spraw związanych z dzieleniem wyrazów do celów składu komputerowego. Jak zostało już wspomniane, czasami ingerencja człowieka staje się niezbędna. Poprawnie zaprojektowany system składu taką ingerencję powinien umożliwiać. Użytkownik powinien mieć w szczególności możliwość wprowadzenia zakazu dzielenia niektórych słów.

W przypadku TEX-a nierozsądne byłoby manipulowanie za każdym razem przy zestawie wzorców. Użytkownik TEX-a może podzielić słowo według własnego uznania na dwa sposoby.

Pierwszy sposób, to umieszczenie go na liście wyjątków, nadrzędnej w stosunku do wzorców. Jest to, niestety, metoda wygodna jedynie w językach niefleksyjnych. W języku polskim umieszczenie jakiegoś słowa na liście wyjątków wiąże się (w przypadku TEX-a) z podaniem od razu wszystkich jego odmian, co jest raczej nieporęczne. Na szczęście problem taki nie pojawia się zbyt często. Głównym powodem tego typu zabiegów bywają słowa złożone lub obcego pochodzenia (bądź i jedno i drugie).

Drugi sposób polega na wskazaniu w tekście dodatkowych miejsc podziału (ang. *discretionary hyphens*). Większość systemów DTP oferuje użytkownikom taką możliwość. W przypadku TEX-a należy ostrożnie posługiwać się tą metodą, bo można przeszkodzić TEX-owi w automatycznym wstawianiu drobnych odstępów między znakami (inaczej podcięć, ang. *implicit kerns*).

TEX oferuje użytkownikowi jeszcze parę innych możliwości, pozwalających na panowanie nad algorytmem podziału wyrazów.

Zabronienie podziału wyrazu jest w TEX-u sprawą trywialną, wystarczy wyraz „zamknąć" w tzw. pudełko (TEX-owa komenda \hbox).

Użytkownik może też określić minimalną długość oddzielonego początku i końca słowa. Domyślnie TEX nie podzieli słowa krótszego niż pięcioliterowe: początek nie może być krótszy niż dwie litery, koniec – nie krótszy niż trzy litery. Przy składaniu w wąskiej szpalcie może się pojawić konieczność złagodzenia tych rygorów, zwłaszcza, że w języku polskim za dopuszczalne uznaje się oddzielanie pojedynczej samogłoski na początku słowa i dwuliterowej sylaby na końcu. Z drugiej strony w składzie wysokiej jakości powinno się unikać dzielenia wyrazów w ogóle, a tym bardziej na zbyt krótkie fragmenty. Możliwość sterowania wielkością dzielonego słowa jest więc bardzo przydatna w praktyce.

Zasady dobrego składu wymagają, aby wyrazy dzielone nie pojawiły się w sąsiednich wierszach i by ostatni wiersz na prawej stronie nie zawierał słowa dzielonego. Składy nie spełniające tych wymogów uważa się (i słusznie) za brzydkie. TEX dostarcza parametrów pozwalających skutecznie utrudniać brzydkie składanie. W szczególności można wręcz zabronić umieszczania słowa dzielonego w ostatnim wierszu na stronie. Może to sprawić pewne kłopoty algorytmowi łamiącemu wiersze na strony, ale to już zupełnie inna historia.

Zostaje jeszcze problem słów zawierających łącznik, np. *biało-czerwony*. W języku angielskim słowa złożone, takie jak *machine-oriented* przenosi się

machine-
-oriented

podczas gdy w języku polskim stanowczo zaleca się dostawienie dodatkowo łącznika na początku następnego wiersza:

biało-
-czerwony

TEX pozwala dołączyć do zestawu komend standardowych komendy definiowane przez użytkownika.

dokończenie na s. 14

ARTUR CHOJNACKI
CEWIS JV ZWUT – Siemens AG
ZBIGNIEW KALISIAK
Instytut Informatyki
Politechnika Warszawska

Uniwersalny monitor szyny dla dydaktycznego systemu mikroprocesorowego

Uruchamianie prototypowych systemów mikroprocesorowych bez odpowiednich narzędzi sprzętowych i programowych jest trudne i mało efektywne. Dostępne przyrządy są drogie, a ich cena często wielokrotnie przewyższa wartość prototypu.

Proces uruchamiania prototypu można umownie podzielić na trzy etapy:

1 – weryfikacja połączeń i sygnałów elektrycznych,
2 – weryfikacja zależności czasowych między sygnałami,
3 – uruchamianie oprogramowania.

Na etapie pierwszym są testowane połączenia między układami w celu wykrycia zwarć i przerw, a także kształty (np. poziomy napięć, szybkość narastania, wypełnienie itp.) sygnałów w systemie. O ile przyrządy stosowane na etapach pierwszym i drugim (oscyloskopy, analizatory stanów logicznych itp.) są uniwersalne i można je stosować niezależnie od struktury systemu prototypowego i elementów z jakich został zbudowany, to uruchamianie oprogramowania, a więc etap trzeci, wymaga narzędzi odpowiednich dla danego procesora. Najczęściej są używane emulatory procesorów, których sercem jest emulowany procesor. Układy towarzyszące pozwalają operatorowi sterować tym procesorem w sposób niezależny od prototypu. Możliwe jest między innymi: załadowanie programu i danych z komputera nadrzędnego do pamięci uruchamianego systemu, wykonywanie programu metodą instrukcja po instrukcji albo aż do zatrzymania, zastawianie pułapek, ujawnianie zawartości rejestrów procesora.

Analogicznie do etapów uruchamiania, można wyróżnić trzy poziomy monitorowania urządzeń mikroprocesorowych:
1 – fizyczny (testowane są parametry fizyczne sygnałów),
2 – logiczny (testowane są zależności między sygnałami cyfrowymi),
3 – poziom testowania oprogramowania.

Poziom trzeci można dodatkowo podzielić na testowanie na poziomie:
- kodu wynikowego,
- mnemoników instrukcji assemblera,
- instrukcji języka wysokiego poziomu.

Dlaczego monitor szyny?

Wyposażenie każdego dydaktycznego stanowiska laboratoryjnego w pełny zestaw narzędzi jest zazwyczaj bardzo kosztowne. Nie jest to także zawsze konieczne.

W dydaktycznym systemie mikroprocesorowym DSM zastosowano konstrukcję modułową [2]. Każdy moduł przechodzi etap pierwszy uruchamiania w czasie produkcji. W trakcie zajęć studenci zazwyczaj nie muszą już testować połączeń i parametrów sygnałów. Oscyloskop wykorzystywany na etapie pierwszym zdaje doskonale egzamin również na etapie zależności czasowych między sygnałami cyfrowymi, jeżeli liczba tych sygnałów nie jest zbyt duża, a taka sytuacja występuje w trakcie uruchamiania projektów studenckich. Oscyloskop jest urządzeniem stosunkowo tanim i uniwersalnym, można więc umieścić go przy każdym stanowisku.

Pozostaje zatem do rozwiązania problem uruchamiania oprogramowania. Emulatory są drogie i wprowadzanie nowych rodzin układów byłoby kosztowne. Opanowanie obsługi nowego przyrządu wymaga także pewnego czasu. Zdecydowano się więc na opracowanie takiego narzędzia, które łączyłoby w sobie cechy analizatora stanów logicznych (możliwość monitorowania dużej liczby sygnałów) z cechami emulatorów (możliwość uruchamiania oprogramowania), a jednocześnie nie było uzależnione od rodzaju użytego w prototypie mikroprocesora.

Oprogramowanie na poziomie kodu wynikowego jest niezależne od typu procesora. Wprawdzie te same kody dla różnych procesorów mogą oznaczać inne instrukcje, jednak nie wpływa to na sposób odczytu i prezentacji, natomiast interpretacja kodu należy do operatora. Zatrzymując się na tym poziomie uruchamiania oprogramowania, można w prosty sposób zrealizować przeznaczone do tego celu urządzenie uniwersalne.

W systemie DSM zastosowano klasyczną strukturę szynową. Zdecydowano więc, że najbardziej odpowiednim urządzeniem będzie monitor szyny. Podstawowymi funkcjami monitora szyny są:
- testowanie i prezentacja stanu szyny,
- wymuszanie dowolnego stanu logicznego na liniach szyny.

Sposób wykorzystania tych funkcji decyduje o funkcjonalności urządzenia.

Koncepcja systemu monitora

Podstawą przyjętej koncepcji była konieczność przystosowywania monitora szyny do mikroprocesora sterującego prototypem. Dla uzyskania łatwości zmiany konfiguracji, prostoty obsługi, uniwersalności i otwartości systemu, wybrano rozwiązanie, w którym monitor szyny jest jednym z pakietów systemu DSM i komunikuje się z programem sterującym,

uruchamianym w komputerze nadrzędnym klasy IBM PC. Do komunikacji zastosowano łącze RS 232C. Realizacja programowa funkcji monitora umożliwia rozwijanie urządzenia bez modyfikacji części sprzętowej.

Konfigurowanie monitora

Standard szyny zakłada „słabą definicję". Określono położenie linii adresowych, danych sterujących transmisją, systemu przerwań oraz arbitrażu szyny, lecz brak jest definicji znaczenia poszczególnych linii. W pełni są określone tylko linie zasilania oraz linie o charakterze diagnostycznym, wykorzystywane do sterowania pracą prototypu.

W czasie konfigurowania monitora pozostałym liniom szyny są przypisywane nazwy użytkownika związane z ich przeznaczeniem. Możliwe jest połączenie linii pojedynczych w grupy wielobitowe (np. stworzenie opisów szyny danych, szyny adresowej), zdefiniowanie typów linii (trójstanowa lub otwarty kolektor) oraz, opcjonalnie, poziomu aktywnego linii pojedynczej.

Zdecydowano, że w przypadku operacji na pamięci systemu i rejestrach układów wejścia-wyjścia, monitor będzie działał jak moduł czynny ubiegający się o zasoby systemu. Konieczne jest więc zdefiniowanie sekwencji strobów generowanych w następujących przypadkach:
- przejęcia szyny systemu,
- zwolnienia szyny systemu,
- operacji odczytu zawartości pamięci lub rejestrów układów we-wy,
- operacji zapisu do pamięci lub rejestrów układów we-wy.

```
TITLE   Z80
PINS
  DATA  [D0]        TS        ;8-bitowa szyna danych
  ADDR  [AD0,AD8]   TS        ;16-bitowa szyna adresowa
  RD    c21         TS LOW    ;strob odczytu
  WR    c22         TS LOW    ;strob zapisu
  MREQ  c19         TS LOW    ;odwołanie do pamieci
  BUSRQ a21         OC LOW    ;żądanie przejęcia szyny
  BDIS  a22         OC LOW    ;odcięcie nadajników szyny
END

TIMINGS                       ;blok definicji sekwencji strobów
 BUSGET                       ;procedura przejecia szyny orzez moduł monitora
   CLRBIT   BUSRQ             ;zapisanie zera do rejestru, do którego została
                              ;przypisana linia BUSRQ
   OPENBUFF BUSRQ             ;aktywacja bufora, do którego przypisana jest
                              ;linia BUSRQ
   DELAY    35                ;opóźnienie 3us*35=105us
   CLRBIT   BDIS              ;wystawienie 0 logicznego na linii BDIS
 END
 BUSFREE                      ;procedura oddania szyny
   SETBIT     BDIS            ;wycofanie strobu odcięcia nadajników
   SETBIT     BUSRQ           ;wycofanie strobu żądania szyny
   RVALBUFF   BDIS,BUSRQ      ;odtworzenie stanów linii
   RACTBUFF   BDIS,BUSRQ      ;odtworzenie aktywności rejestru
 END
 SPACE 1 MEMORY 8 'Memory'    ;definicja przestrzeni adresowej typu pamięć
  WRITE                       ;procedura zapisu do pamięci
    SETBIT     MREQ,RD,WR     ;wycofanie sygnałów MRQ,WR,RD
    OPENBUFF   MREQ,RD,WR     ;aktywacja bufora linii strobów
    OPENBUFF   ADDR           ;aktywacja buforów linii adresowych
    OPENBUFF   DATA           ;aktywacja bufora linii danych
   LABEL                      ;początek pętli sekwencyjnego zapisywania
                              ;danych
    PUTADDR    ADDR           ;wystawienie adresu na szynę systemu
    PUTDATA    DATA           ;wystawienie danych na szynę systemu
    CLRBIT     MREQ           ;odwołanie do pamięci
    CLRBIT     WR             ;generacja strobu zapisu
    SETBIT     WR             ;wycofanie strobu zapisu
    SETBIT     MREQ           ;wycofanie strobu odwołania do pamięci
    NEXTADDR                  ;wyliczenie następnego adresu
   LOOP                       ;koniec pętli
    RVALBUFF   DATA           ;odtworzenie stanu rejestru bufora danych
    RACTBUFF   DATA           ;odtworzenie aktywności bufora danych
    RVALBUFF   ADDR           ;odtworzenie stanu rejestrów buforów adresu
    RACTBUFF   ADDR           ;odtworzenie aktywności rejestrów buforów
                              ;adresu
    RVALBUFF   MREQ,RD,WR     ;odtworzenie stanu rejestru bufora strobów
    RACTBUFF   MREQ,RD,WR     ;odtworzenie aktywności rejestru bufora strobów
  END                         ;procedury zapisu
 END
END
```

Rys. 1. Fragment przykładowego pliku konfiguracji monitora szyny

W celu tworzenia konfiguracji monitora, opracowano specjalny język opisu oraz kompilator konfiguracji BUSCOMP. Przygotowany w postaci pliku ASCII opis konfiguracji należy skompilować. Plik wynikowy jest czytany w czasie inicjalizacji modułu monitora oraz programu sterującego monitorem szyny – MONITOR. Rys. 1 przedstawia fragment przykładowego opisu konfiguracji monitora szyny dla procesora Z80.

Komunikacja z operatorem

Pakietem monitora szyny steruje program MONITOR. Wykorzystuje on technikę menu i okien do prezentacji stanu monitora i uruchamianego systemu. Jest możliwe używanie zarówno klawiatury, jak i myszy.

Konfiguracja monitora jest wczytywana w czasie inicjalizacji programu. Definicje linii są wykorzystywane do przedstawiania stanu szyny systemu, natomiast procedury opisujące sekwencje strobów są przekazywane do modułu monitora i umieszczane tam w pamięci RAM.

Opracowano specjalny protokół transmisji między programem nadrzędnym, a modułem monitora, który umożliwia:
- wykrywanie błędów oraz ich korekcję (przez ponowienie nadawania),
- wykrywania przerwania połączenia (np. po wyłączeniu zasilania w kasecie systemu),
- rekonfigurację modułu monitora, po włączeniu zasilania w kasecie systemu.

Na listwie maskującej modułu monitora umieszczono diody elektroluminescencyjne wskazujące stan monitora (gotowość do odbioru danych, nadanie, tryb pracy).

Operacje na liniach szyny systemowej

Użytkownik ma możliwość wymuszenia dowolnego stanu logicznego na wybranej linii. Ponieważ jednak moduł monitora nie jest jedynym modułem, który może wymuszać stany na szynie, istnieje możliwość wystąpienia kolizji wymuszeń. Kolizje mogą doprowadzić do uszkodzenia elementów systemu. Sprzętowa realizacja wykrywania kolizji wymaga użycia, oprócz rejestru wyjściowego, rejestru referencyjnego i komparatora. Z tego powodu zrezygnowano z takiego rozwiązania zastępując je programowym porównywaniem stanu szyny i zawartości rejestrów wyjściowych monitora. Wykrycie kolizji powoduje odłączenie rejestrów monitora od szyny systemu. Zdefiniowany typ linii (trójstanowa, otwarty kolektor) określa algorytm wykrywania kolizji.

W systemie DSM zdefiniowano linie priorytetu szeregowego o połączeniu łańcuchowym. Każdy moduł ma linię wejściową priorytetu szeregowego PCI oraz wyjściową PCO. Moduł monitora umożliwia:
- połączenie logiczne linii PCI z linią PCO,
- ustawianie dowolnego stanu na linii PCO.

Sterowanie pracą uruchamianego systemu

Na szynie systemu przewidziano następujące trzy linie sygnałowe służące do sterowania pracą systemu i wykonywaniem programów:
- linię inicjalizacji systemu – *RESET*,
- linię włączenia układu pracy krokowej – *SMODE*, gdzie aktywne moduły z zaimplementowaną funkcją pracy krokowej reagują na uaktywnienie tej linii przejściem do trybu pracy krokowej,
- linię sterowania pracą krokową –*STEP* (zbocze opadające zezwala na zakończenie bieżącego cyklu transmisji).

Odpowiednie operowanie tymi liniami pozwala sterować pracą systemu prototypowego. Na szczególną uwagę zasługuje sposób realizacji zatrzymania wykonywania programu. W systemie DSM pułapką określa się mechanizm zatrzymywania

działania programu, którym może sterować operator. Realizacja sprzętowa układu pułapkowania wymaga użycia rejestru wartości, rejestru maski oraz komparatora. Ze względu na dużą liczbę potrzebnych układów scalonych zrezygnowano z takiego rozwiązania. Zamiast tego wprowadzono tryb powolnej prasy systemu, w którym moduł czynny jest utrzymywany w trybie pracy krokowej, a monitor generuje na linii *–STEP* impulsy kończące kolejne cykle. W każdym takim cyklu transmisji jest porównywany stan szyny systemu z definicjami aktywnych pułapek. Jeżeli stan ten odpowiada dowolnej definicji, monitor, nie kończąc bieżącego cyklu transmisji, kończy generować impulsy na linii *–STEP*, wyświetlając odpowiedni komunikat.

Operacje na zasobach systemu

Operacje na zasobach systemu są dostępne tylko wtedy, gdy w pliku opisującym konfigurację monitora szyny zdefiniowano odpowiednie sekwencje strobów. Użytkownik może zdefiniować do czterech przestrzeni adresowych, opisując w każdej procedurę zapisu i procedurę odczytu. Zdefiniowanie tych procedur umożliwia odczyt i modyfikację zawartości pamięci i rejestrów wejścia-wyjścia. Monitorowanie na bieżąco zasobów systemu wymaga jednak odczytu, nie tylko na żądanie użytkownika, lecz także np. po wykonaniu kolejnego cyklu transmisji procesora. Odczyt pamięci nie powoduje zmiany jej stanu, natomiast odczyt rejestru układu we-wy może spowodować zmianę stanu automatu wewnętrznego (automat taki sprawia np. to, że kolejne odczyty rejestru pod tym samym adresem powodują w rzeczywistości odczyt danych z różnych rejestrów). Z tego powodu zdefiniowano dwa typy przestrzeni adresowych (definiując przestrzeń adresową określa się jej typ):

- przestrzeń adresową typu pamięć (MEMORY), w której jest możliwy odczyt zawartości bez ingerencji użytkownika,
- przestrzeń adresową typu wejście-wyjście (I/O), w której odczyt i zapis mogą nastąpić tylko na żądanie użytkownika.

Ograniczenia

W projekcie dążono do uproszczenia części sprzętowej. Z tego powodu zrezygnowano ze sprzętowej realizacji:

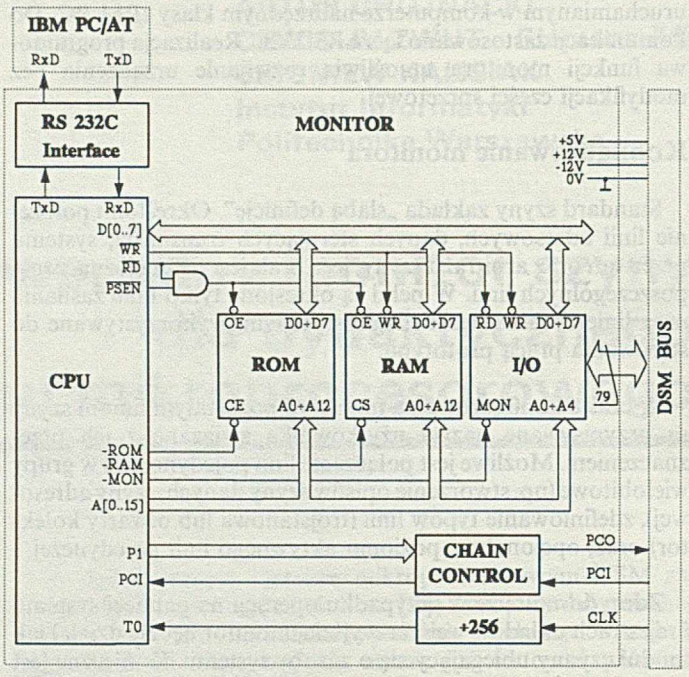
- układu pułapkowania,
- układu wykrywania kolizji wymuszeń,
- zbierania śladu,
- kontroli napięć zasilania w monitorowanym systemie (wymagałoby to zastosowania niezależnego zasilania monitora).

Odczyt i zapis stanu wewnętrznego mikroprocesorów jest realizowany przez wykonanie odpowiednich procedur. Procedury te nie powinny być jednak umieszczane w pamięci uruchamianego systemu. W czasie wykonywania tych procedur należy odłączyć od szyny procesora pamięć systemu, a na jej miejsce umieścić pamięć z zapisanymi procedurami (lub np. monitor szyny, który może wymuszać kolejne kody instrukcji na szynie), a następnie wykonać te procedury krokowo. Budowa modułów biernych w systemie DSM nie zakłada jednak ich buforowania, nie wszystkie moduły czynne zawierają także układy pracy krokowej. Z tego powodu zrezygnowano z funkcji odczytu i zapisu stanu mikroprocesora prototypu.

Budowa modułu monitora szyny

Schemat blokowy monitora szyny dla systemu DSM przedstawia rys. 2. Podstawowym elementem monitora jest blok mikroprocesora – „CPU" – zawierający mikrokomputer jednoukładowy typu 8051 oraz bufory sterujące szyną danych i szyną adresową monitora.

Blok „RS 232C Interface" zapewnia dopasowanie napięciowe



Rys. 2. Schemat blokowy pakietu monitora szyny

sygnałów iTxD i iRxD w standardzie TTL do standardu elektrycznego łącza RS 232C (V24).

Mikrosterownik 8051 ma rozdzieloną przestrzeń adresową programu i danych. W przestrzeni programu umieszcza się pamięć EPROM z zapisanym programem sterującym. W przestrzeni danych najczęściej jest umieszczana pamięć RAM, w której są przechowywane dane używane w czasie działania programu. W celu umożliwienia wykonywania procedur, zdefiniowanych przez użytkownika i zapamiętywanych w pamięci RAM, połączono przestrzenie adresowe programu i danych w jedną. Funkcję tę realizuje bramka *AND* widoczna na rys. 2.

Blok „Chain Control" steruje liniami priorytetu szeregowego PCI i PCO. Sygnał zegara systemowego jest podawany do dzielnika częstotliwości „÷252". Częstotliwość sygnału wyjściowego zostaje zmierzona w bloku mikrokontrolera.

W bloku „I/O" znajduje się dziesięć rejestrów wejściowo-wyjściowych (dalej nazywanymi buforami), umożliwiających odpowiednio: odczyt stanu szyny oraz wymuszanie odpowiedniego stanu na szynie. Użyto typowych układów scalonych zawierających podwójne rejestry 8-bitowe (74HCT652 dla linii trójstanowych oraz 74ALS654 dla linii typu „otwarty kolektor"). Konsekwencją takiego rozwiązania była konieczność połączenia sąsiadujących linii szyny w wiązki 8-bitowe. Każda taka wiązka jest przypisana do określonego bufora a wszystkie linie należące do niej są jednocześnie aktywne lub nieaktywne. Jak wykazała praktyka, w większości przypadków ograniczenie to nie sprawia kłopotów, gdyż odpowiednie położenie strobów na liniach przypisanych do różnych buforów pozwala zapobiec wystąpieniu ewentualnych konfliktów. Dodatkowym efektem tego rozwiązania jest także to, że minimalną jednostką przydzielaną do grup wielobitowych może być jeden bufor, czyli osiem bitów.

Program sterujący pracą monitora szyny – MONITOR

Program MONITOR pośredniczy w komunikacji między użytkownikiem a systemem DSM. Dzięki wykorzystaniu techniki menu i okien jest on łatwy do opanowania i prosty w użyciu.

Prezentowany program jest aplikacją środowiska Turbo Vision. Środowisko to jest znane z kompilatorów firmy Borland, takich jak Turbo Pascal 6.0, Borland C++. Z tego też powodu konstrukcja ekranu i wybór funkcji odbywa się w sposób charakterystyczny dla tego środowiska. Ekran monitora jest podzielony na trzy części:
- wiersz menu (wiersz w górnej części ekranu),
- obszar okien (centralna część ekranu),
- wiersz stanu (wiersz w dolnej części ekranu).

Wiersz menu zawiera nazwy grup poleceń. W obszarze okien są umieszczane okna zawierające obraz stanu poszczególnych elementów systemu uruchamianego (np. szyny, pamięci, definicje pułapek itp.). Wiersz stanu zawiera, w czasie operacji na menu, krótkie objaśnienia zleceń lub opisy aktywnych klawiszy.

Grupa poleceń *File*

Grupa ta zawiera następujące polecenia:
- tworzenia nowego pliku ASCII – *New*,
- czytania z dysku wcześniej utworzonego pliku – *Open*,
- zmiany nazwy pliku – *Save as*,
- zapisywania pliku – *Save*,
- zmiany bieżącego katalogu – *Change dir*,
- tymczasowego wyjścia do systemu DOS – *Dos shell*.

Grupa poleceń *Edit*

Zawiera takie podstawowe polecenia edytora tekstowego, jak:
- odtworzenie ostatnio skasowanego ciągu znaków – *Undo*,
- kopiowanie – *Paste*,
- kasowanie – *Cut*,
- wstawianie do ClipBoard – *Copy*,
- czyszczenie Clipboard – *Clear*.

Grupa poleceń *Search*

Pozwala wyszukiwać i wymieniać wzorce w edytowanym pliku.

Grupa poleceń *Windows*

Umożliwia zmianę rozmiarów i położenia wyświetlanych okien, aktywację wybranego okna lub jego usunięcie z ekranu.

Grupa poleceń *Monitor*

Jest charakterystyczna dla programu MONITOR. Zawiera dwa polecenia:
- prezentacji stanu szyny systemu i statusu monitora – *Bus State*,
- prezentacji stanu rejestrów wyjściowych modułu monitora – *Outputs*.

Jeżeli w czasie konfigurowania monitora zdefiniowano przestrzenie adresowe, to w menu pojawiają się polecenia dodatkowe. Dla przestrzeni adresowej typu pamięć:
- *Load* ..., polecenie realizujące zapisanie kodu programu wynikowego do pamięci uruchamianego systemu,
- *Open* ..., wyświetlające w odpowiednim oknie obraz zawartości wybranego obszaru pamięci.

W przypadku przestrzeni typu wejście-wyjście pojawia się tylko polecenie *Open* ... otwierające bufor w pamięci komputera nadrzędnego do realizacji operacji na rejestrach. Nazwy poleceń są uzupełniane (w miejscu wystąpienia trzech kropek ...) nazwami nadanymi poszczególnym przestrzeniom adresowym w opisie konfiguracji.

Grupa poleceń *Run*

Grupa ta dotyczy uruchamiania programów w systemie prototypowym. Dostępne są następujące polecenia:
- inicjalizacja systemu – *Reset system*,
- zakończenie cyklu maszynowego – *Step*,
- praca powolna, jest sprawdzane wystąpienie pułapki – *Slow Run*,
- praca animowana, jest sprawdzane wystąpienie pułapki – *Animate*,
- praca w czasie rzeczywistym – Free Run, monitor zostaje odłączony od szyny systemu, a wystąpienie pułapek nie jest sprawdzane.

Zatrzymanie programu następuje albo po wykryciu przez monitor wystąpienia pułapki, albo po wydaniu przez operatora polecenia *Step*.

Grupa poleceń *Breakpoints*

Realizuje obsługę pułapek programowych. Dostępne są pułapki na liniach użytkownika, z możliwością maskowania wybranych linii oraz pułapka na linii wejściowej priorytetu szeregowego PCI, z możliwością określenia aktywnego zbocza. Obsługa pułapek obejmuje: definiowanie, aktywację i deaktywację pułapek.

* * *

Autorzy, przed przystąpieniem do realizacji projektu, uczestniczyli w zajęciach laboratoryjnych w Instytucie Informatyki Politechniki Warszawskiej, w ramach których wykorzystywano Modularny System Mikroprocesorowy MSM [4] i związany z nim monitor szyny. Zebrane doświadczenia w dużej mierze zadecydowały o cechach funkcjonalnych przedstawionego urządzenia i jego ostatecznym kształcie. Realizacja programowa systemu pułapkowania nie sprawia większych kłopotów w laboratorium studenckim. Nie jest to jednak rozwiązanie wygodne w przypadku uruchamiania systemów profesjonalnych. Obecnie w Instytucie Informatyki PW autorzy artykułu opracowują nowy model monitora szyny, w którym bufory serii HCT i ALS zostaną zastąpione układami LCA firmy XILINX [3]. Duża złożoność układów tego typu pozwala zrealizować sprzętową pułapkę na całej szerokości szyny z możliwością maskowania pojedynczych linii, bez drastycznego rozbudowania urządzenia. Usunięta zostanie także konieczność łączenia linii w wiązki. Każdą linią będzie można sterować osobno oraz dowolnie definiować jej typ.

Kolejne wersje oprogramowania monitora zostaną uzupełnione o:
- rozwinięty system objaśnień kontekstowych,
- mechanizm wywoływania zewnętrznych narzędzi programowych (asemblerów, konsolidatorów itp.),
- polecenie prezentacji zawartości pamięci w postaci instrukcji asemblera odpowiedniego procesora,
- procedury autotestowania pakietu monitora,
- procedury diagnostyczne systemu.

Konstrukcja monitora szyny, w postaci pakietu systemu DSM, zakłada jego stosowanie w tym systemie. Jest to jednak urządzenie na tyle uniwersalne, że może znaleźć zastosowanie w dowolnym systemie o strukturze szynowej. Przykładem może tu być specjalna wersja monitora, wykonana na zamówienie Wydziału Informatyki Politechniki Białostockiej dla systemu MSM [4]. W systemie tym zastosowano pełną definicję szyny, a oprogramowanie zostało przystosowane do obowiązującego standardu.

Jak uruchamiać programy?

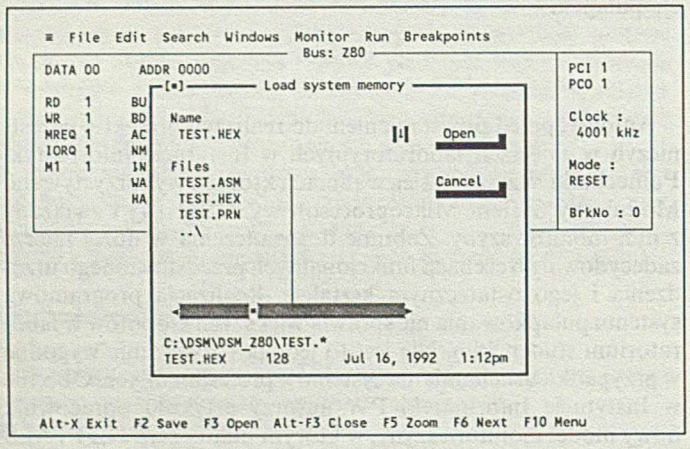
Przykładowy program dla procesora Z80:

```
        LD  HL, 30H ;dane z obszaru o adresie 30h
        LD  DE, 40H ;należy skopiować do obszaru o adresie 40h
        LD  BC, 16  ;długość bloku 16 bajtów
        LDIR        ;kopiowanie
        HALT        ;zatrzymanie procesora
INF:    JP  INF     ;pętla nieskończona
```

zapisano w pliku *TEST.ASM* i skompilowano. Otrzymano plik *TEST.HEX* zawierający kod wynikowy programu. Do uruchomienia programu użyto:

- pakietu procesora Z80,
- pakietu pamięci 8-bitowej,
- odpowiednio skonfigurowanego pakietu monitora szyny.

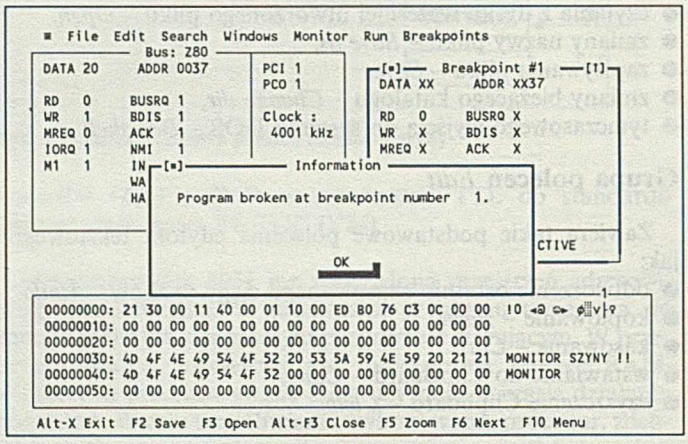
Po uruchomieniu programu MONITOR sterującego pakietem monitora, należy załadować do pamięci systemu kod wynikowy.



Na ekranie: okno dialogowe do wybierania z listy plików, pliku zawierającego kod programu, w tle – okno prezentujące stan szyny i monitora.

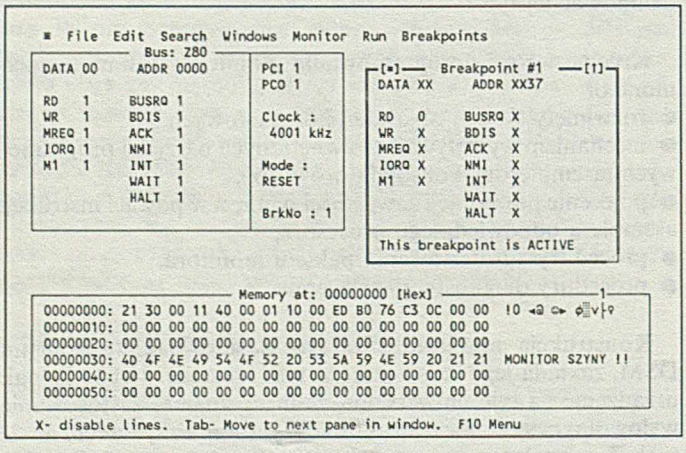
W programie nie zdefiniowano zawartości obszaru danych, wpisano więc dane bezpośrednio pod adres 30*h* wykorzystując okno obrazu pamięci. W celu zatrzymania programu w trakcie kopiowania danych, zdefiniowano pułapkę na odczycie bajtu o adresie 37*h*.

Wystąpienie pułapki sprawdzane jest tylko w trybach *Slow Run* i *Animate*. Po zatrzymaniu wykonywania programu monitor przechodzi do trybu *Step* oraz wyświetla numer wykrytej pułapki.

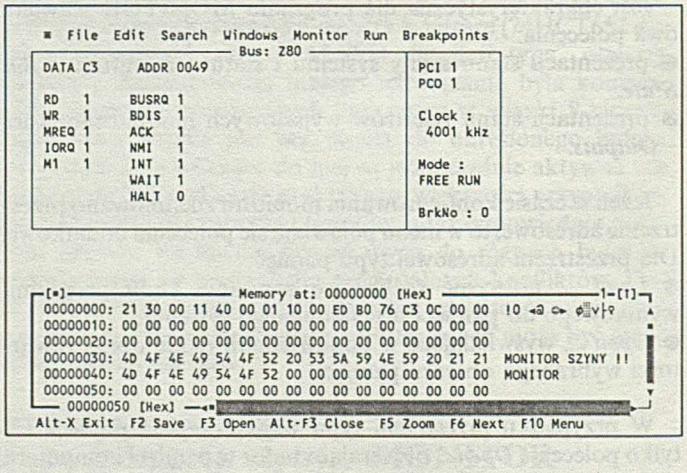


Na ekranie: okno dialogowe zawierające numer wykrytej pułapki, okno stanu szyny przedstawia aktualny jej stan, a okno pamięci zawartość pamięci po skopiowaniu siedmiu bajtów.

Dalej uruchomiono program w trybie *Free Run*. Procesor zatrzymał się na instrukcji *HALT*, czekając na przerwanie lub inicjalizację systemu.

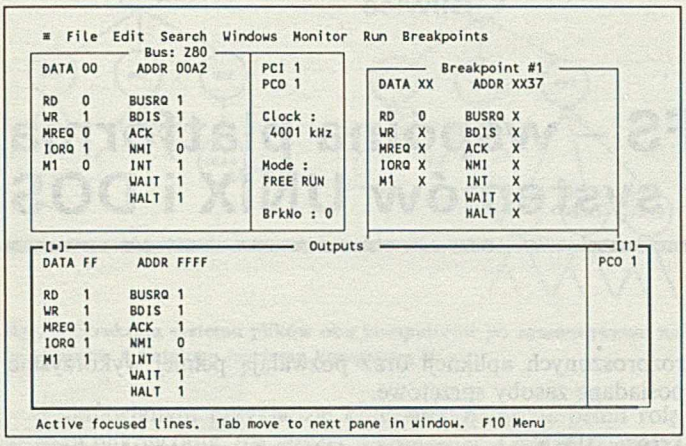


Na ekranie: u góry z lewej – okno prezentujące stan szyny i monitora (**Bus: Z80**), u góry z prawej – okno definicji pułapki (**Breakpoint # 1**), u dołu – okno przedstawiające zawartość pamięci systemu (**Memory at: 00000000 [HEX]**).



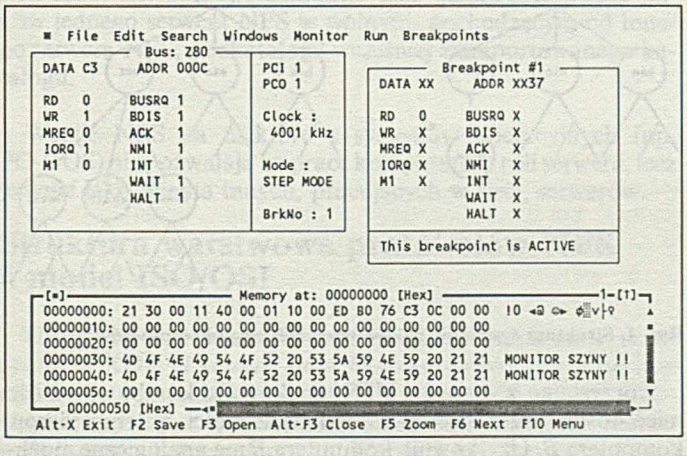
Na ekranie: monitor jest odłączony od szyny systemu (**Mode: Free Run**), stan linii *HALT* (zero logiczne) wskazuje na zatrzymanie procesora.

Wymuszając na linii *NMI* stan zero, wygenerowano przerwanie niemaskowane – procesor kontynuuje wykonywanie programu.



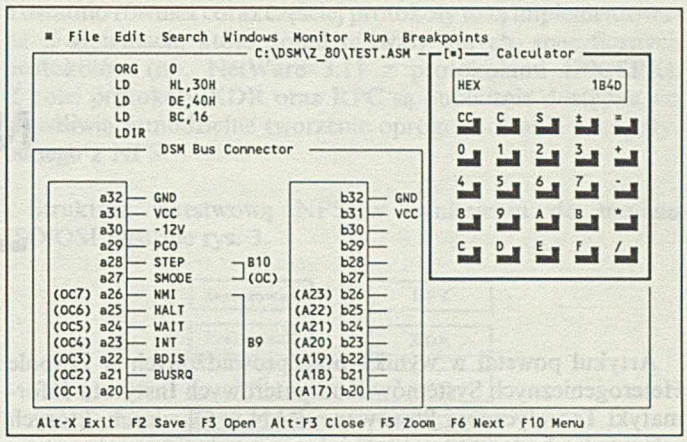
Na ekranie: u dołu – okno rejestrów wyjściowych (**Outputs**), na linii *NMI* stan zero, u góry – w oknie stanu szyny, stan linii *HALT* wskazuje, że procesor kontynuuje pracę.

Po zatrzymaniu wykonywania pętli nieskończonej, można sprawdzić efekt programu.



Na ekranie: monitor szyny w trybie pracy krokowej, obraz okna zawartości pamięci został odświeżony.

Jeśli zachodzi taka potrzeba, użytkownik może skorzystać z wbudowanego edytora, sprawdzić definicję szyny, a nawet coś policzyć używając kalkulatora dwusystemowego.



Na ekranie: prawy górny róg – kalkulator, dolny lewy róg – definicja szyny, w tle – edytor.

* * *

Autorzy pragną podziękować wszystkim pracownikom Instytutu Informatyki PW, dzięki którym mogli opracować i wykonać przedstawiony monitor szyny. Autorzy dydaktycznego systemu mikroprocesorowego – Jerzy Chrząszcz i Grzegorz Mazur dali nam całkowitą swobodę w prowadzeniu projektu. Kierownik zespołu Marek Pawłowski, nasz opiekun naukowy, pomagał i dopingował, gdy zaszła taka potrzeba. Henryk Jeż weryfikował dokumentację, a Romuald Siwik i Krzysztof Wacławski pomogli nam w montażu.

LITERATURA

[1] Chojnacki A., Kalisiak Z.: Uniwersalny Monitor Szyny dla Dydaktycznego Systemu Mikroprocesorowego. Praca dyplomowa, IIPW, 1992

[2] Chrząszcz J.R., Mazur G.B.: Laboratoryjny System Mikroprocesorowy. INFORMATYKA nr 6, 1992

[3] The Programmable Gate Array Data Book. Xilinx, San Jose, California 1991

[4] Woźniak A., Rzeszut J., Sosnowski J.: Modularny System Mikroprocesorowy. Raport Badawczy IIPW nr 34, Warszawa 1981.

Zakład Wydawniczy ELEKTRONIZACJA wydaje w I kwartale 1993 r.
bardzo poszukiwany na rynku, obszerny i nowoczesny

**Słownik skrótów angielskich
stosowanych w elektronice, informatyce i telekomunikacji**

Słownik możesz otrzymać pocztą pod wskazany adres, wpłacając preferencyjną kwotę 50.000 zł, w terminie do końca grudnia 1992 r., na konto: PBK III O/Warszawa nr 370015-1573-139-11, Zakład Kolportażu Wydawnictwa SIGMA-NOT sp. z o.o., 00-950 Warszawa, skr. poczt. 1004. Zamawiając słownik w przedpłacie, masz gwarancję otrzymania go po cenie ulgowej, bez wysokiej marży sieci sprzedaży i kosztów wysyłki.

GRZEGORZ KOWALCZYK
Instytut Informacji Teoretycznej
i Stosowanej
PAN
Gliwice

NFS – wspólna platforma systemów UNIX i DOS

Artykuł powstał w wyniku prac prowadzonych w Zespole Heterogenicznych Systemów Komputerowych Instytutu Informatyki Teoretycznej i Stosowanej PAN w Gliwicach, których celem było zbadanie rozwiązań integrujących różnorodne systemy komputerowe przez sieć typu LAN. Wśród rozpatrywanych rozwiązań znajdował się sieciowy system plików NFS (*Network File System*).

System NFS pozwala komputerom połączonym siecią typu LAN (najczęściej są stosowane Ethernet bądź Token Ring, rzadziej wolniejsze łącze szeregowe z protokołem SLIP – *Serial Line Internet Protocol*) na wzajemny dostęp do indywidualnych zasobów dyskowych, przy czym współpracujące komputery mogą mieć różne architektury oraz pracować pod kontrolą różnych systemów operacyjnych. Użytkownik może korzystać z plików znajdujących się na dyskach komputerów pracujących w sieci i udostępnianych mu przez system NFS, tak jak gdyby pliki te fizycznie znajdowały się na dysku maszyny, na której on pracuje (dostępne są przez podanie nazwy ścieżkowej). W ten sposób może on przechowywać zarówno pliki danych, jak i programy. Programy wykonywane przez użytkownika na własnym komputerze mogą być ładowane z dysku odległej maszyny, a w trakcie wykonywania mogą współpracować z udostępnianą przez NFS przestrzenią dyskową.

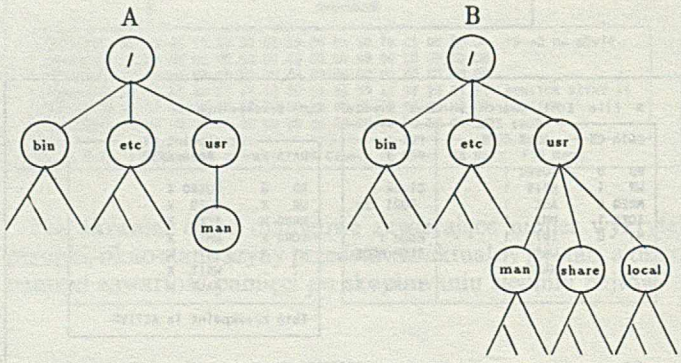
System NFS pozwala wielu różnym komputerom korzystać z tych samych plików, powodując obniżenie kosztów składowania danych, oraz umożliwiając realizację sieciowych baz danych. Mimo, iż na wstępie podkreślono, że NFS pracuje w sieciach lokalnych, nic nie stoi na przeszkodzie użycia jakichkolwiek innych rozwiązań sieci. Różnice mogą wystąpić jedynie w szybkości pracy systemu.

System NFS został opracowany przez firmę Sun Microsystems w 1985 r. jako główny element koncepcji sieci otwartych ONC (*Open Network Computing*). Od tego czasu kod źródłowy tego systemu został udostępniony wielu producentom komputerów, którzy dokonali implementacji NFS w swoich produktach. Według ocen firmy Sun Microsystems na koniec 1990 r. ok. 2,2 mln systemów komputerowych, począwszy od komputerów klasy PC, a kończąc na komputerach o dużej mocy obliczeniowej, korzystało z NFS. Obecnie wersje NFS są dostępne dla większości najczęściej stosowanych systemów operacyjnych, jak np.: UNIX, SunOS, VMS, Ultrix-32, ALX/370, PC-DOS, OS/2, HP-UX, MacOS, AmigaDOS oraz wielu innych. Dostępne są również rozszerzenia NFS na sieciowe systemy operacyjne, np. NetWare NFS.

Wszystkie wersje NFS, dostosowane do szerokiej gamy komputerów i systemów operacyjnych, mogą ze sobą współpracować przez sieć, tworząc środowisko dostosowane do realizacji rozproszonych aplikacji oraz pozwalają pełniej wykorzystać posiadane zasoby sprzętowe.

Idea NFS

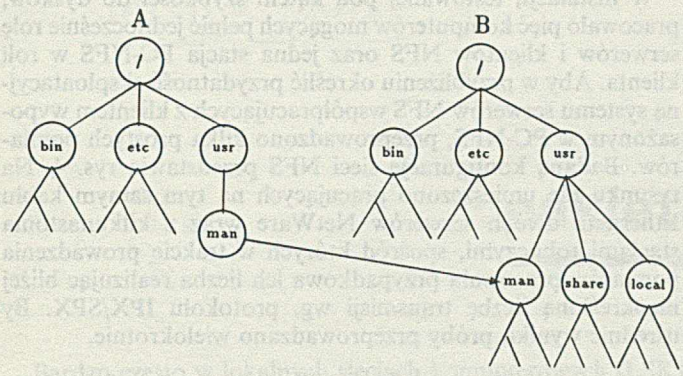
By zobrazować ideę NFS rozpatrzmy sytuację przykładową. Przyjmijmy, że w sieci istnieją dwa komputery, na których jest uruchomiony NFS: komputer *A* i komputer *B*, pracujące pod kontrolą systemu operacyjnego UNIX. Systemy plików na obu komputerach mają strukturę przedstawioną na rys. 1.



Rys. 1. Struktura systemów plików przykładowych komputerów

Korzystając z systemu NFS użytkownik komputera może mieć dostęp przez sieć do całości bądź części systemu plików komputera *B*. Użytkownik komputera *B* ma analogiczne możliwości wykorzystania systemu plików komputera *A*. Załóżmy, że użytkownik komputera *A* chce skorzystać z plików zawartych w katalogu */usr/man* komputera *B*. Administrator komputera *B* musi udostępnić żądany katalog, przez umieszczenie ścieżki dostępu do tego katalogu w pliku zawierającym listę katalogów, które będą widoczne w sieci (katalogów eksportowanych) i wykonując odpowiednie polecenie systemu – udostępnić zawartość katalogu sieci. Od tego momentu będą one widoczne w sieci aż do zamknięcia usług komputera *B* bądź zamknięcia NFS. Eksportując dany katalog zachowuje się drzewiastą strukturę systemu plików, dzięki czemu użytkownikom sieci automatycznie udostępnia sią podkatalogi położone poniżej eksportowanego katalogu. Z kolei administrator komputera *A* musi umieścić wyeksportowany katalog (bądź któryś z jego podkatalogów) w dowolnym pustym katalogu swojego systemu plików, w omawianym przykładzie w katalogu */usr/man* (w systemach nie będących pochodnymi UNIX-a może odbywać się to w nieco inny, właściwy dla tych systemów, sposób. Jednocześnie będą dostępne katalogi leżące poniżej zamontowanego katalogu w hierarchicznej strukturze eksportowanej

przez komputer *B* części systemu plików. Schematycznie sytuację przedstawia rysunek 2.



Rys. 2. Struktura systemu plików obu komputerów po zamontowaniu na komputerze A katalogu /usr/man komputera B

W omawianym przykładzie komputer *B* będzie pełnił rolę serwera plików (ang. *file server*), a komputer *A* będzie klientem (ang. *client*). W rzeczywistości sytuacja może być bardziej złożona, ponieważ każdy z UNIX-owych komputerów może być jednocześnie serwerem i klientem, tzn. może udostępniać swoje pliki innym komputerom i montować części systemów plików innych komputerów, przy czym może być jednocześnie klientem wielu serwerów NFS.

Należy zaznaczyć, że komputer może eksportować jedynie zawartość swojego własnego systemu plików (nie może reeksportować katalogu zamontowanego z innego serwera NFS), natomiast jest możliwe zamontowanie zdalnego systemu plików jednego serwera NFS w wolnym, pochodzącym od innego serwera NFS, podkatalogu wcześniej zaimportowanego katalogu.

Wersje NFS na niektóre z systemów operacyjnych (np. PC-DOS) nie pozwalają na pracę komputera w roli serwera, lecz jedynie jako klienta innych, pracujących w sieci, serwerów.

Struktura warstwowa protokołów NFS a model ISO/OSI

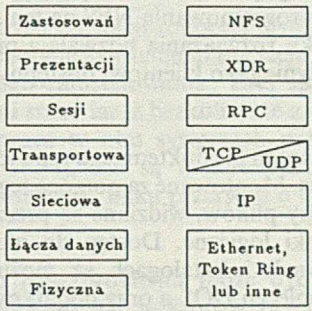
Usługi NFS w warstwie prezentacji są realizowane przez protokół XDR (*External Data Representation*), dostarczający niezależną od architektury metodę reprezentacji danych. Aplikacje korzystające z XDR mają możliwość wymiany danych w środowisku heterogenicznym. Warstwę sesji tworzy protokół RPC, będący zestawem systemowo niezależnych operacji umożliwiających wykonywanie procedur na odległych komputerach, stanowiąc środowisko do tworzenia rozproszonych aplikacji w systemach heterogenicznych.

Warstwa transportowa zawiera protokóły TCP i UDP. Protokół TCP (*Transmission Control Protocol*) został zastosowany w sieci ARPANET. Udostępnia on bezbłędne połączenie wirtualne między dwoma stacjami w trybie *full duplex*. Drugi z protokołów warstwy transportowej – UDP (*User Datagram Protocol*) – jest używany przez NFS najczęściej. Jest to protokół bezpołączeniowy, realizujący komunikację między stacjami za pomocą tzw. datagramów. Korzystające z tego protokołu aplikacje muszą same korygować błędy powstale w trakcie transmisji.

Protokołem warstwy sieciowej jest protokół IP (*Internet Protocol*), wywodzący się również z sieci ARPANET. Jego zadaniem jest podział transmitowanej informacji na pakiety, wybór drogi między węzłami oraz tworzenie sprzęgów międzysieciowych. Warstwy łącza danych oraz fizyczna są realizowane w zależności od rodzaju użytkowanej sieci lokalnej.

Grupa protokołów TCP/IP stanowi integralną część większości implementacji systemu UNIX i systemów pochodnych, a ostatnio również coraz częściej protokoły te są implementowane w systemach, które dotąd używały innych, specyficznych protokołów (np. NetWare 3.11 z protokołami IPX/SPX). Z kolei protokoły XDR oraz RPC są publicznie dostępne, co umożliwia samodzielne tworzenie oprogramowania kompatybilnego z NFS.

Strukturę warstwową NFS w odniesieniu do modelu ISO/OSI ilustruje rys. 3.



Rys. 3. Model warstwowy NFS w odniesieniu do modelu ISO/OSI

UNIX-owe serwery NFS

Powszechne wykorzystanie systemu operacyjnego UNIX na komputerach o różnych architekturach oraz mnogość implementacji NFS na ten system, stwarza duże prawdopodobieństwo, że użytkownicy systemów komputerowych spotkają się z serwerem NFS pracującym na systemie operacyjnym UNIX.

W działającej w naszym Instytucie instalacji serwerami NFS były następujące komputery: VAX Station II/RC z systemem Ultrix-32, SUN SPARCStation 1+ i SUN SPARCStation IPC (oba pod systemem SunOS 4.1.1) oraz ALR PowerPro z procesorem Intel 486 pod kontrolą systemu INTERACTIVE UNIX 2.2 (wszystkie wymienione systemy operacyjne są pochodnymi UNIX-a).

Uruchamianie NFS na wszystkich tych systemach przebiegało w podobny sposób. Dla systemów Ultrix-32 oraz INTERACTIVE UNIX, NFS jest oprogramowaniem opcjonalnym, które należy załadować na wcześniej zainstalowany system operacyjny, natomiast w wypadku systemu SunOS, NFS jest ładowany w trakcie instalacji systemu operacyjnego. Po załadowaniu oprogramowania konieczne było utworzenie systemowych baz danych, zawierających informację o sieci, dostępnych w sieci serwerach oraz o udostępnianych sieci częściach systemu plików (pliki baz systemowych znajdują się w katalogu */etc*, a najważniejsze z nich, to pliki *hosts* i *exports*) i następnie zainicjowanie pracy NFS. W przypadku systemu INTERACTIVE UNIX administrator komputera ma ułatwione zadanie, ponieważ może posłużyć się programem administratora systemu (*sysadm*) w celu załadowania oprogramowania, utworzenia baz danych i uruchomienia NFS. Szczegółowe informacje na temat instalacji NFS są zawarte w dokumentacji dostarczanej wraz z produktem, jednakże użytkownik mający doświadczenie w instalacji i uruchamianiu NFS na jednym z systemów UNIX-owych nie powinien napotkać większych trudności w wypadku innej wersji UNIX-a.

W instalacji pracującej w Instytucie Informatyki Teoretycznej i Stosowanej PAN w Gliwicach, wszystkie uruchomione na

maszynach UNIX-owych wersje NFS współpracowały ze sobą bez zastrzeżeń, zarówno jako serwery plików, jak i klienci dostępnych w sieci serwerów NFS.

IBM PC a środowisko NFS

Komputery klasy IBM PC można wykorzystać w środowisku NFS w dwojaki sposób. Pierwszym sposobem jest zainstalowanie na PC jednej z dostępnych wersji UNIX-a (np. INTERACTIVE UNIX bądź SCO UNIX), dla których istnieją pełne implementacje NFS, pozwalające komputerom tego typu na pracę w charakterze zarówno serwerów, jak i klientów sieciowego systemu plików. Drugim rozwiązaniem, bardziej interesującym z punktu widzenia większości użytkowników komputera IBM PC działającego pod systemem operacyjnym PC DOS, jest zainstalowanie oprogramowania NFS na ten system. Jednakże istniejące na rynku rozwiązania pozwalają pracować takiemu komputerowi jedynie jako klientowi dostępnych w sieci serwerów NFS.

Najbardziej znanym produktem tego typu jest PC-NFS firmy Sun Microsystems. Montowane za pomocą tego oprogramowania zdalne systemy plików, widziane są przez użytkonika PC jako kolejne dyski logiczne. Dostęp do nich oraz wszelkie operacje na plikach i katalogach są możliwe przy użyciu standardowych poleceń DOS-a oraz dodatkowych programów pakietu PC-NFS, takich, jak np. RCP bądź FTP. Współpracujący w sieci użytkownicy PC-NFS mogą korzystać ze wspólnych plików zgromadzonych na serwerze NFS. Współdzielenie plików jest możliwe również między użytkownikami PC-NFS, a użytkownikami innych, wyposażonych w NFS systemów. Oprogramowanie PC-NFS pozwala na wykonywanie na dyskach logicznych, będących częściami systemów plików serwerów NFS, większości poleceń DOS-a (w wersji 3.0 bądź późniejszej) prócz następujących poleceń: *CHKDSK, DISKCOMP, DISKCOPY, FDISK, FORMAT, JOIN, LABEL, RECOVER, SUBST* oraz *SYS*. Dla organizowania wspólnego dostępu do plików i rekordów jest dostępne jedynie DOS-owskie polecenie *SHARE*, ale wykorzystanie usług NETBIOS-a jest niemożliwe.

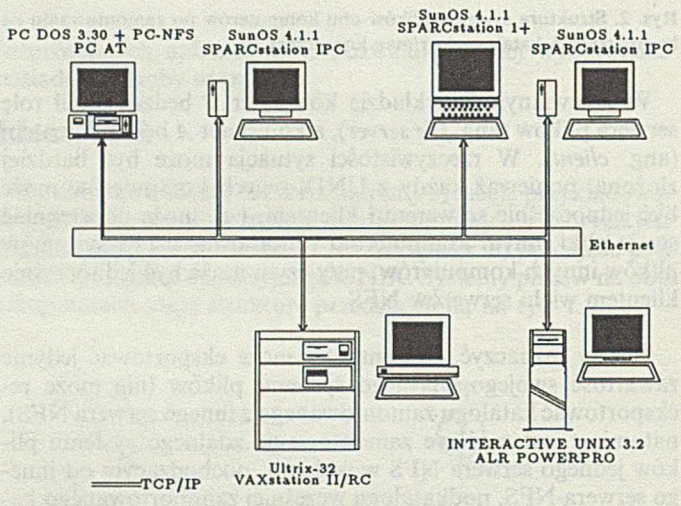
Pakiet PC-NFS umożliwia również korzystanie z drukarek sieciowych, realizację poczty elektronicznej oraz wykorzystanie przez sieć taśmowego zapisu archiwalnego (ang. *backup*) zainstalowanego na jednym z dostępnych w sieci serwerów NFS. Możliwe jest również zdalne włączanie się do dostępnych w sieci, a nie będących serwerami NFS, komputerów i korzystanie z ich zasobów stosując programy pakietu PC-NFS o nazwach FTP (*File Transfer Protocol*), RSH (*Remote Shell*) oraz TELNET. Program FTP pozwala m.in. na kopiowanie plików w obu kierunkach, RSH umożliwia zdalne wykonywanie poleceń bądź programów, natomiast TELNET tworzy z komputera IBM PC terminal znakowy typu vt100.

Standardem połączenie sieciowego jest Ethernet, lecz jest możliwe także wykorzystanie łącza szeregowego.

Dogodność korzystania z PC-NFS zakłóca fakt występowania między systemami UNIX-owymi a DOS-em różnic w składniach nazw plików. Nazwy UNIX-owe muszą być poddawane konwersji na format DOS-owski, przez co ulegają one czasem zniekształceniu do stopnia utrudniającego identyfikację pliku. Aby tego uniknąć, należałoby współdzielone przez NFS pliki nazywać zgodnie z wymogami DOS-a, które to nazwy są podzbiorem zbioru nazw dopuszczalnych przez UNIX.

Uwagi eksploatacyjne

W instalacji, testowanej pod kątem szybkości do dysków, pracowało pięć komputerów mogących pełnić jednocześnie rolę serwerów i klientów NFS oraz jedna stacja PC-NFS w roli klienta. Aby w przybliżeniu określić przydatność eksploatacyjną systemu serwerów NFS współpracujących z klientem wyposażonym w PC-NFS, przeprowadzono kilka prostych pomiarów. Badaną konfigurację sieci NFS przedstawia rys. 4. Na rysunku nie umieszczono pracujących na tym samym kablu Etherneta dwóch serwerów NetWare wraz z kilkunastoma stacjami roboczymi, spośród których w trakcie prowadzenia pomiarów pracowała przypadkowa ich liczba realizując bliżej nieokreśloną liczbę transmisji wg. protokołu IPX/SPX. By uśrednić wyniki, próby przeprowadzano wielokrotnie.



Rys. 4. Schemat badanej sieci NFS

Próby polegały na pomiarze czasu kopiowania trzech zbiorów znakowych o wielkości 332 kB każdy, między dyskami poszczególnych serwerów NFS, a dyskiem lokalnym klienta, którym był komputer ALR VEISA 386 o zegarze 33 MHz, z zainstalowanym PC-NFS. Wyniki mogą dać jedynie przybliżony obraz jakości badanego systemu NFS, tak więc dla porównania przeprowadzono analogiczne pomiary dla pracującej na tym samym kablu i w tych samych warunkach sieci NetWare, która cechuje się stosunkowo dużą szybkością dostępu do plików. Jako stacji roboczej NetWare użyto tego samego komputera, jak w wypadku pomiarów dla PC-NES. Wyniki prób zawiera poniższa tabela.

Wyniki badań sieci NFS w porównaniu z siecią NetWare

Typ serwera	Czas odczytu z serwera	Czas zapisu na serwer	Wydajność odczytu	Wydajność zapisu
Serwery NFS:				
ALR Power Pro	12 s	37 s	83 kB/s	26.91 kB/s
Sun Sparc Stat. 1+	9 s	17 s	110.66 kB/s	58.58 kB/s
Sun IPC	8 s	19.5 s	124.5 kB/s	51.07 kB/s
VAX Station II	11 s	26 s	90.54 kB/s	38.3 kB/s
Serwer NetWare'a: IMC 386 25MHz	6.5 s	4.5 s	153.2 kB/s	221.33 kB /s

dokończenie na s. 14

Czy zaprenumerowaliście Państwo INFORMATYKĘ na rok 1993?

MIROSŁAW SZCZYPIŃSKI
Instytut Sterowania i Techniki Systemów
Politechnika Wrocławska

Współpraca pakietu PC NFS z siecią Novell

Bardzo często w lokalnych sieciach komputerowych (LSK) zdarza się, że sieć Ethernet integruje serwery i stacje robocze pracujące pod kontrolą różnych systemów operacyjnych. Jednym z bardziej popularnych rozwiązań jest umieszczanie w tej samej LSK serwerów systemów UNIX i NetWare. Oba systemy wymagają uruchomienia na stacjach roboczych programów umożliwiających dostęp do serwerów sieci, przy czym programy te są specyficzne dla danego typu systemu operacyjnego.

Jednym z popularniejszych pakietów programowych, umożliwiających dostęp do zasobów serwera systemu UNIX, jest PC NFS. W sieci Novell na stacji roboczej instaluje się programy standardowo dołączane z systemem. Inicjacja PC NFS wymaga także rezydentnego zainstalowania. Taki sposób inicjowania pracy z serwerami sprawia, iż bezkolizyjne zainstalowanie obu pakietów oraz praca w obu systemach przy wykorzystaniu jednej karty sieciowej, nie są możliwe. Po prostu pakiety programów wykorzystują kilka przerwań o tym samym numerze.

Rozwiązaniem tego problemu może być przechowywanie i odtwarzanie poprawnych wektorów przerwań zależnie od

```
AUTOEXEC.BAT:
echo off
SET PATH=a:\;a:\NFS
cls
echo ************************************************************
echo Wybierz: 1 dla rownoczesnej pracy z UNIX-em i siecia NOVEL
echo                    2 praca wylacznie z UNIX-em
echo ************************************************************
key 12
if errorlevel 50 goto x1
if errorlevel 49 goto x0
goto exit
:x0
echo Wybrano: 1
SET NFSDRIVE=a:
NFSRUN
echo off
set path=a:\;a:\nfs
set comspec=a:\nfs
set nfsdrive=a:
a:\nfs\f1.exe
a:\novell\ipx
a:\novell\net5
i:
login
goto exit
:x1
echo Wybrano: 2
SET NFSDRIVE=a:
NFSRUN
echo off
set path=a:\;a:\nfs
set comspec=a:\nfs
set nfsdrive=a:
:exit
echo on

/* Programy f1.c i f2.c . Kompilacja w Turbo C */
#include<stdio.h>
#include<conio.h>
#include<dos.h>
FILE *OF;
void interrupt far (*Ref1)();
void interrupt far (*Ref2)();
void interrupt far (*Ref3)();
void interrupt far (*Ref4)();
void interrupt far (*Ref5)();

void ex(void)
{
 printf("\n\r\a\a\a%s\a\a\a\n", "Blad ");
 exit(1);
}

void main(void)
{
  printf("\n\r      >>==Foto1==<<\n\r");
  Ref1=getvect(0x08);
  Ref2=getvect(0x09);
  Ref3=getvect(0x14);
  Ref4=getvect(0x21);
  Ref5=getvect(0x28);
  OF=fopen("vect1","w"); /* w programie f2.c : OF=fopen("vect2","w"); */
  if(OF->flags==_F_ERR) ex();
  fprintf(OF,"%lx\n",Ref1);
  fprintf(OF,"%lx\n",Ref2);
  fprintf(OF,"%lx\n",Ref3);
  fprintf(OF,"%lx\n",Ref4);
  fprintf(OF,"%lx\n",Ref5);
  if(fclose(OF)<0) ex();
}

/* Program r1.c i r2.c .  Kompilacjja w Turbo C */
#include<stdio.h>
#include<dos.h>
FILE *OF;
void interrupt far (*Ref1)();
void interrupt far (*Ref2)();
void interrupt far (*Ref3)();
void interrupt far (*Ref4)();
void interrupt far (*Ref5)();

void ex(void)
{
 printf("\n\r\a\a\a%s\a\a\a\n\r\n", "Blad ");
 exit(1);
}

void main(void)
{  printf("\n\r      >>==Re1Fo==<<\n\r");
  OF=fopen("vect1","r");    /* w r2.c tu musi byc OF=fopen("vect2","r");*/
  if(OF->flags==_F_ERR) ex();
  fscanf(OF,"%lx\n",&Ref1);
  fscanf(OF,"%lx\n",&Ref2);
  fscanf(OF,"%lx\n",&Ref3);
  fscanf(OF,"%lx\n",&Ref4);
  fscanf(OF,"%lx\n",&Ref5);
  setvect(0x08,Ref1);
  setvect(0x09,Ref2);
  setvect(0x14,Ref3);
  setvect(0x21,Ref4);
  setvect(0x28,Ref5);
  if(fclose(OF)<0) ex();
}
```

kontekstu wykonywanych operacji. Poniżej przedstawiono postać zbioru AUTOEXEC.BAT uruchamianego na stacji roboczej oraz wydruki programów pomocniczych w najprostszej wersji.

Przedstawione oprogramowanie jest wykorzystywane do współpracy z systemami NetWare 3.10 i 3.11, SunOS oraz Open Desktop w Laboratorium Zespołu Systemów Sterowania Instytutu Sterowania i Techniki Systemów Politechniki Wrocławskiej. Stacje sieciowe są wyposażone w karty Ethernet Plus. Przed wykonaniem programu z pakietu PC NFS należy wykonać program *r1.exe*, dla sieci Novell – *r2.exe*. Obraz wektora przerwań dla PC NFS jest zapamiętany po wykonaniu programu *f1.exe*, a dla sieci Novell – *f2.exe* (program ten należy uruchomić natychmiast po rozpoczęciu sesji w sieci Novell). Do prawidłowej pracy należy zadbać o zachowanie kolejności ładowania programów (patrz AUTOEXEC.BAT). Łatwo zauważyć, że rozwiązanie to można zastosować w wielu przypadkach wystąpienia konfliktu wektorów obsługi przerwań.

Ze świata

SYSTEC'92

Międzynarodowe Targi oraz Kongres SYSTEC'92 (20–23.10.1992 Monachium) były okazją do trzeciego już spotkania producentów i użytkowników systemów zbudowanych z wykorzystaniem protokołu komunikacyjnego dla zautomatyzowanych systemów wytwarzania MAP (*Manufacturing Automation Protocol*). Tematyce tej patronuje organizacja EMUG (*European MAP/TOP Users Group*), która wspólnie z Sekcją Inżynierii Produkcji Stowarzyszenia Inżynierów Niemieckich VDI-ADB zorganizowała na terenie Targów ekspozycję tematyczną pt. „MAP/MMS w działaniu" oraz znaczną część obrad Kongresu.

MAP jest protokołem zdefiniowanym przez grupę użytkowników zrzeszonych w organizacji WFMUG (*World Federation of MAP/TOP Users Group*) jako standard, oparty na modelu ISO systemów otwartych OSI (*Open System Interconnection*), ale przeznaczony do budowy sieci łączących w jeden spójny system komputery i urządzenia automatyki. Z punktu widzenia użytkownika i projektanta zautomatyzowanych systemów wytwarzania istotne są oferowane przez ten standard usługi pozwalające ujednolicić język MMS (*Manufacturing Message Specyfication*), w którym następuje wymiana danych między poszczególnymi zadaniami.

Jednym z ważniejszych akcentów tegorocznego SYSTEC była wspomniana już ekspozycja tematyczna „MAP/MMS w działaniu" zlokalizowana w hali nr 16. Demonstrowała ona wspólną sieć łączącą komputery 23 niezależnych wystawców. Obok takich potentatów, jak AEG, Allan-Bradley, DEC czy Bull, swoje wyroby prezentowały również mniejsze firmy, jak PROCOS, SISCO, Cycle, Reflex. Wśród demonstrowanych rozwiązań znajdowały się sterowniki numeryczne, oprogramowanie dla komputerów osobistych, stanowisko robota przemysłowego, systemy do testowania MAP oraz MMS. Firma Allen Bradley pokazała całą celę produkcyjną, w której wszystkie sterowniki połączono siecią MAP.

Na uwagę zasługuje fakt, że użyty sprzęt i oprogramowanie były bardzo różnorodne i pochodziły od wielu producentów.

dokończenie na III s. okładki

Tam gdzie minus oznacza dzielenie

dokończenie ze str. 3

W MEX-u została zdefiniowana komenda \=, którą należy podczas przygotowywania tekstu umieścić w miejscu łącznika. Kontynuując przykład, *biało-czerwony* należałoby podać TEX-owi jako *biało\=czerwony*. Użytkownik musi pamiętać o konsekwentnym przestrzeganiu tej zasady, a o resztę zadba już TEX. Pozostawienie zwykłego łącznika oznacza słowo niepodzielne. Jest to przydatne w przypadku takich słów, jak np. K-202, których oczywiście dzielić nie należy.

* * *

Jak widać, zagadnienie automatycznego dzielenia wyrazów to problem bardzo obszerny. Wchodzą tu w grę zagadnienia językoznawcze, typograficzne i informatyczne.

Praktyka pokazuje, że bardzo efektywny algorytm automatycznego podziału słów, w połączeniu z niekoniecznie bardzo wygodnymi możliwościami ingerowania ręcznego, pozwala użytkownikowi zapanować w pełni nad procesem dzielenia słów przez system komputerowego składu tekstów.

Wydaje się nam, że rozwiązania zastosowane przez Knutha są na tyle uniwersalne (a do tego sprawdzone w praktyce), że warto je szerzej rozpropagować. Mogą się okazać szczególnie przydatne w Polsce, gdzie rynek software'owy dopiero powstaje i dobre nomen omen wzorce są rzeczą nie do pogardzenia.

NFS – wspólna platforma systemów UNIX i DOS

dokończenie ze str. 12

Jak już wspomniano wcześniej, NFS umożliwia realizację sieciowych baz danych. Dla programisty, od strony DOS-a jest dostępna funkcja *SHARE*, a w środowisku UNIX-a – funkcja *lockf*. Obie umożliwiają „zamknięcie" dostępu do rekordu w pliku.

Podsumowując możemy stwierdzić, że różnice w szybkości realizacji dostępu do dysków w wypadku sieci NetWare na najszybszych z przebadanych serwerów NFS nie są tak znaczne (1,5 do 4 razy) i dlatego uwzględniając stosunkowo niskie koszty (w wypadku posiadania wcześniej zakupionego serwera UNIX-owego), kształtujące się w granicach 300 dolarów na stację roboczą PC-NFS, w wielu wypadkach uzasadniają wybór tego rozwiązania, pozwalając jednocześnie pełniej wykorzystać posiadany sprzęt i umożliwiając współpracę użytkowników wielu systemów operacyjnych.

LITERATURA

[1] INTERACTIVE NFS Guide. INTERACTIVE Systems Corporation, 1990
[2] NetWare Buyer's Guide, March 1991, Inc., 1991
[3] ONC/NFS. A Technology Guide to Distributed Computing. Sun Microsystems, Inc., 1992
[4] PC-NFS (Version 3.0.1). Sun Microsystems, Inc. 1989
[5] SunOS 4.1.1 vol. System and Network Administration. Sun Microsystems, Inc., 1990
[6] ULTRIX-32 Network Management Guide. Digital Equipment Corporation, 1987.

Spis treści rocznika INFORMATYKA 1992

Artykuły problemowe

	nr	s.
Propozycja strategii rozwoju informatyki i jej zastosowań w Rzeczypospolitej Polskiej		
Cz. 1	1	1
Cz. 2	2	1
Cz. 3	3	1
BARANOWSKI JAN – Od BTRIEVE-a do SQl-a. Nowoczesne, wielopoziomowe środowisko baz danych firmy NOVELL	3	15
BARTA JANUSZ, MARKIEWICZ RYSZARD – Podstawowe problemy prawa komputerowego	1	8
BIEDALAK KRZYSZTOF, WOŹNIAK PIOTR – Metoda SuperMemo – optymalizacja procesu uczenia się	10	1
BIZACKI PIOTR, PALIWODA GRAŻYNA, SKOMOROWSKI MAREK – Interakcyjny system do projektowania i analizy cyfrowych układów logicznych	11	5
BORSA MARIUSZ – Nowe techniki w konstruowaniu interfejsów użytkownika	3	12
BYRSKA MAŁGORZATA – Wytyczne EWG w sprawie ochrony programów komputerowych a polski projekt prawa autorskiego	7	17
CABAN DARIUSZ, JARNICKI JACEK – Rekonfigurowalny filtr obrazów cyfrowych	6	15
CHOJNACKI ARTUR, KALISIAK ZBIGNIEW – Uniwersalny monitor szyny dla dydaktycznego systemu mikroprocesorowego	12	4
CHRZĄSZCZ JERZY R., MAZUR GRZEGORZ B. – Laboratoryjny system mikrokomputerowy	6	1
CUPISZ ROMAN – LINC – Kompletne środowisko CASE	2	21
DĄBROWSKA IWONA, DEPOWSKA DOROTA – Metody automatycznego rozpoznawania znaków	4	19
DEPOWSKA DOROTA, DĄBROWSKA IWONA – Metody automatycznego rozpoznawania znaków	4	19
FRYŹLEWICZ ZBIGNIEW, UŁASIEWICZ JĘDRZEJ – Scheduler transakcyjny dla zastosowań komunikacyjnych	2	10
FUGLEWICZ PIOTR – Zabawa w słówka, czyli kilka informacji o językoznawstwie statystycznym	3	20
GAIK DARIUSZ, KOTULA JACEK, PIECHA JAN – System autorski ASYS 3.0	5	17
ISZKOWSKI WACŁAW — Informatyzacja sukcesem biznesu	8	2
JACKOWSKI BOGUSŁAW, RYĆKO MAREK – Tam gdzie minus oznacza dzielenie	12	1
JERCZYŃSKA MARIA – Badania krajowego rynku komputerowego	4	23
KALISIAK ZBIGNIEW, CHOJNACKI ARTUR – Uniwersalny monitor szyny dla dydaktycznego systemu mikroprocesorowego	12	4
KIETZMAN BOGDAN – Komputerowe układy sterowania numerycznego	4	10
KOLEŚNIK KRYSTYNA – Koncepcja symulacji obiektowej z wykorzystaniem języka Turbo Pascal 5.5	4	14
KOSMOWSKA-MISZALSKA DANUTA – Strojenie mechanizmu rozumowania systemów ekspertowych z uwzględnieniem warunków niepewności	9	17
KOTULA JACEK, GAIK DARIUSZ, PIECHA JAN – System autorski ASYS 3.0	5	17
KOWALCZYK GRZEGORZ – NFS – wspólna platforma systemów UNIX i DOS	12	10
KOWALSKI KAZIMIERZ – Wykorzystanie sieci komputerowych w dydaktyce	4	1
KOWALSKI PIOTR – T^3 – program przetwarzania wielojęzycznych tekstów naukowo-technicznych	5	1
KRAJEWSKI RAFAŁ – Dlaczego NIS?	11	19
KUBIAK GRZEGORZ, SZEWC ANDRZEJ – Reprezentacja danych w zautomatyzowanych systemach informacji legislacyjnej	1	13
LANGE KATARZYNA – Praktyczne wykorzystanie protokołu NetBIOS	7	12
LASEK MIROSŁAWA – Doświadczenia realizacji systemu oceny sytuacji przedsiębiorstwa	10	19
MADRJAS KRZYSZTOF – AD/Cycle. IBM-owska filozofia inżynierii projektowania oprogramowania	8	18
MICHALEWSKI EDWARD – DIANA-9 – pakiet wspomaganej komputerowo analizy diagnostycznej i projektowania struktur organizacyjnych	11	9
MICHALIK KRZYSZTOF, TWARDOWSKI ZBIGNIEW – System PC-SHELL: narzędzie do budowy dziedzinowych systemów ekspertowych	10	23
MARKIEWICZ RYSZARD, BARTA JANUSZ – Podstawowe problemy prawa komputerowego	1	8
MARZEC BOGDAN – OS-9 – siła i słabość standardu	5	13
MAZUR GRZEGORZ B., CHRZĄSZCZ JERZY R. – Laboratoryjny system mikrokomputerowy	6	1

	nr	s.
ORYLSKA JADWIGA – Dydaktyka informatyki w uczelniach rolniczych	11	22
OWCZARKOWSKI RYSZARD, SZUKALSKI STANISŁAW – Ekspertory system wspomagania decyzji w planowaniu makroenergetycznym	10	10
PALIWODA GRAŻYNA, BIZACKI PIOTR, SKOMOROWSKI MAREK – Interakcyjny system do projektowania i analizy cyfrowych układów logicznych	11	5
PIECHA JAN, GAIK DARIUSZ, KOTULA JACEK – System autorski ASYS 3.0	5	17
PIECHA JAN, PRZYBYLSKI ANDRZEJ, ZYGUŁA JAROSŁAW – APW – graficzne narzędzia programowania	6	20
PIÓRKOWSKA TERESA, PUTERNICKI DARIUSZ, STANISZKIS WITOLD – Inżynieria systemów informatycznych – od strategii do realizacji	8	9
PODRECKI PAWEŁ – Ochrona prawna topografii układów scalonych	7	22
PRZETACZNIK WIESŁAW – Graficzne stacje robocze SPARCstation	4	5
PRZYBYLSKI ANDRZEJ, PIECHA JAN, ZYGUŁA JAROSŁAW – APW – graficzne narzędzia programowania	6	20
PUTERNICKI DARIUSZ, PIÓRKOWSKA TERESA, STANISZKIS WITOLD – Inżynieria systemów informatycznych – od strategii do realizacji	8	9
RYĆKO MAREK, JACKOWSKI BOGUSŁAW – Tam gdzie minus oznacza dzielenie	12	1
STABROWSKI MAREK M. – AMIL – interakcyjny język symulacji układów dynamicznych	11	1
STANISZKIS WITOLD, PIÓRKOWSKA TERESA, PUTERNICKI DARIUSZ – Inżynieria systemów informatycznych – od strategii do realizacji	8	9
STĄPOR KATARZYNA – Strukturalna analiza i projektowanie systemów informatycznych – metodyka Gane'a-Sarsona		
Cz. 1	2	24
Cz. 2	3	25
STĄPOR KATARZYNA, WAWRZYŃCZYK MAREK – Wpływ algorytmów tablic rozproszonych na efektywność działania systemów ekspertowych	9	20
STAWICKI JERZY – BB_POL – narzędzie do budowy tablicowych systemów ekspertowych	10	14
SKOMOROWSKI MAREK, PALIWODA GRAŻYNA, BIZACKI PIOTR – Interakcyjny system do projektowania i analizy cyfrowych układów logicznych	11	5
SKRZYPEK JERZY, SZUBRA MARIUSZ – Symulacyjne gry decyzyjne jako narzędzie aktywnego kształcenia menedżerów	9	12
SURMA JERZY – Modelowanie wiedzy w metodyce KADS	12	15
SZCZYPIŃSKI MIROSŁAW – Współpraca pakietu PC NFS z siecią Novell	12	13
SZEWC ANDRZEJ, KUBIAK GRZEGORZ – Reprezentacja danych w zautomatyzowanych systemach informacji legislacyjnej	1	13
SZEWCZYK AGNIESZKA – Diagnostyka systemów informacyjnych		
Cz. 1. Problemy podstawowe	11	12
Cz. 2. Model diagnostyki systemu informacyjnego	12	22
SZUBRA MARIUSZ, SKRZYPEK JERZY – Symulacyjne gry decyzyjne jako narzędzie aktywnego kształcenia menedżerów	9	12
SZUKALSKI STANISŁAW, OWCZARKOWSKI RYSZARD – Ekspertowy system wspomagania decyzji w planowaniu makroenergetycznym	10	10
SZYJEWSKI ZDZISŁAW – Systemy otwarte, moda czy konieczność?	7	8
SZYJEWSKI ZDZISŁAW – Od metodologii tworzenia systemów informatycznych do CASE	8	1
ŚREDNIAWA MAREK – Sieci Inteligentne – usługi i architektura	9	1
TWARDOWSKI ZBIGNIEW, MICHALIK KRZYSZTOF – System PC-SHELL: narzędzie do budowy dziedzinowych systemów ekspertowych	10	23
UŁASIEWICZ JĘDRZEJ, FRYŹLEWICZ ZBIGNIEW – Scheduler transakcyjny dla zastosowań komunikacyjnych	2	10
WAWRZYŃCZYK MAREK, STĄPOR KATARZYNA – Wpływ algorytmów tablic rozproszonych na efektywność działania systemów ekspertowych	9	20
WINKLER T. – Środowisko systemów CAD	11	25
WOŹNIAK PIOTR, BIEDALAK KRZYSZTOF – Metoda SuperMemo – optymalizacja procesu uczenia się	10	1
WRYCZA STANISŁAW – Metody analizy i projektowania systemów w praktyce	9	26
WRZESZCZ ZDZISŁAW – Inteligentny system gromadzenia informacji faktograficznej	6	9
WRZESZCZ ZDZISŁAW – Operacje wyszukiwania w bazie danych systemu GFG	12	18
WYROSTEK PIOTR – Arytmetyka wektorowa komputera CONVEX C120	1	19
ZIELIŃSKI KRZYSZTOF – Stan i perspektywy rozwoju uczelnianej sieci komputerowej Akademii Górniczo-Hutniczej	7	1
ZONENBERG DARIUSZ — Emulator układowy mikroprocesorów jako narzędzie uruchomieniowe w ujęciu programistycznym	2	15
ZYGUŁA JAROSŁAW, PIECHA JAN, PRZYBYLSKI ANDRZEJ – APW – Graficzne narzędzia programowania	6	20

Z kraju

Fenomen rynku krajowego	1	III okł.
Instytut Maszyn Matematycznych szkoli użytkowników PC	2	8
II krajowa konferencja metodyki nauczania przedmiotów informatycznych w uczelniach rolniczych	3	III okł.
Rozstrzygnięcie konkursu PTI	5	28
Konkurs im. Jerzego Trybulskiego	8	27

Ze świata

	nr	s.
Rozwój pamięci typu „flash" – oprac. *Jan Ryżko*	1	24
Współpraca Intel – IBM	2	13
Informacje z firmy Microsoft	2	23
A jednak Polska uczestniczy w CeBIT'92	2	32
Perspektywa systemów 64-bitowych – *Jan Ryżko*	3	30
Mikroprocesory z zegarem o częstotliwości 100 MHz – *J.R.*	4	25
Mikroprocesory 32-bitowe – *J.R.*	4	25
Światłowody w sieciach komputerowych – *J.R.*	4	25
Wzrost produkcji układów ASIC oraz programowanych – *J.R.*	4	25
Informatyka w Singapurze – *Jan Ryżko*	6	28
Prognozy sprzedaży i zastosowań laserów – *Jan Ryżko*	7	26
Różne architektury RISC – *Jan Ryżko*	10	26
Porozumienie firm Cray i Sun – *J.R.*	10	27
Firma Oracle zainteresowana UNIX-em – *J.R.*	10	27
Nowe rozwiązania firmy Next – *J.R.*	10	27
Wyniki konkursu SOFTARG'92	10	27

Konferencje

CEBIT'92	1	7
Kapsch AG	4	28
Informatyka bankowa	5	23
IV Wyższa Górska Międzynarodowa Szkoła PTI pn. „Współczesne obszary zastosowań informatyki"	6	26
Information Systems Developers Workbench	7	21
Systemy Otwarte	9	11

/unix

Integracja systemów operacyjnych MS-DOS i UNIX – *Paweł Żak*		
Cz. 1	1	27
Cz. 2	2	29
Cz. 3	3	27
W UNIX-ie ostrożność popłaca – *Piotr Carlson*	4	27
Atrapa mechanizmu wyjątków dla C – *Jan Walasek*	5	21
UNIX nad Zatoką San Francisko – *Jacek Surma*	5	24
Jakim jesteś unixowcem? – *Agnieszka Pietrusińska*	6	27

Listy

Szanowna Redakcjo! – *Jacek Przepióra*	2	III okł.

Nowe książki

Wydawnictwa Naukowo-Techniczne	7	7
Wydawnictwa Naukowo-Techniczne	10	25

TERA Spółka Z O. O.

Przedsiębiorstwo Popierania Postępu TERA Spółka z o.o. uprzejmie informuje, że posiadając kilkuletnie doświadczenie w instalacji systemów wspomagających zarządzanie

o f e r u j e :

- opracowanie projektów systemów,
- optymalny kosztowo i rozwojowo dobór sprzętu i oprogramowania,
- instalację systemu u klienta,
- bezpłatnie przez rok konserwację oprogramowania oraz serwis sprzętu wraz z doradztwem techniczno-eksploatacyjnym,
- szybką dostawę uzupełnień konfiguracji lub sprzętu mikrokomputerowego niezależnego od systemu, w tym mikrokomputerów ALR firmy Wearnes Technology.

Wszelkie dodatkowe informacje uzyskają Państwo codziennie oprócz niedziel w Biurze Handlowym

40-025 Katowice, ul. Wojewódzka 31
tel. (faks): 155-26-72, teleks: 315448 tera pl

PAMIĘTAJ! Instalacje XENIX/NOVELL/PC MOS 386 oraz serwis to nasza specjalność – ZAPRASZAMY, ponieważ czterech lat doświadczeń nigdzie nie kupisz.

0/19/90

meditronik SPÓŁKA z o.o.

UMC UNITED MICROELECTRONICS CORPORATION

HEWLETT PACKARD COMPONENTS

BOURNS

- CZĘŚCI ELEKTRONICZNE
- KOMPUTERY PS/1,PS/2
- DRUKARKI HP
- INSTALACJE SIECI KOMPUTEROWYCH

Partnerzy handlowi: ANALOG DEVICES, ITT, MOTOROLA, SAMSUNG, TELEFUNKEN i inni

PRZEDSTAWICIELSTWO

- UKŁADY PAMIĘCI
- UKŁADY KOMPUTEROWE
- UKŁADY KOMUNIKACYJNE I KOMERCYJNE

- TRANSOPTORY
- WSKAŹNIKI ŚWIETLNE
- WYŚWIETLACZE LED
- PRODUKTY KODÓW KRESKOWYCH
- KONTROLERY I CZUJNIKI RUCHU
- TECHNIKA ŚWIATŁOWODOWA
- ELEMENTY W.CZ. I MIKROFALOWE
- PODZESPOŁY DO MONTAŻU POWIERZCHNIOWEGO (SMD)

DYSTRYBUCJA

- POTENCJOMETRY TRIMPOT
- HYBRYDY REZYSTOROWE
- REZYSTORY SUBMINIATUROWE
- BEZPIECZNIKI MULTIFUSE
- POTENCJOMETRY PRECYZYJNE
- POTENCJOMETRY PANELI CZOŁOWYCH I KODERY
- CEWKI I TRANSFORMATORY
- CZUJNIKI CIŚNIENIA, POŁOŻENIA I PRZYŚPIESZENIA

PMI

DYSTRYBUCJA

IBM Business Partner

meditronik sp. z o.o.

00-194 Warszawa,ul. Dzika 4
tel. (02) 6352263, 6352264
fax (02) 6352195, tlx 816075

SO/575/91

JUNISOFTEX Sp. z o.o.

44-100 GLIWICE ul. Konstytucji 11,
tel.-faks 31-75-10, 31-90-81 do 88 w. 250, 272, 282, teleks 036233

JUNISOFTEX – to firma z tradycjami i najdłuższymi doświadczeniami w eksploatacji wielodostępnych systemów komputerowych Novell w kraju.

JUNISOFTEX – to najstarszy w kraju wykonawca własnych wielodostępnych systemów pracujących w sieci NetWare firmy Novell.

JUNISOFTEX – to autoryzowany reseller amerykańskiej firmy Novell.

JUNISOFTEX – to dostawca zintegrowanych systemów komputerowych działających w kilkudziesięciu firmach obejmujących w każdym przedsiębiorstwie, niezależnie od formy własności i dziedziny gospodarki: finanse, majątek obrotowy, majątek trwały i nietrwały, sprzedaż, techniczne przygotowanie produkcji oraz kadry-płace.

JUNISOFTEX – to dostawca i wykonawca autoryzowanych sieci lokalnych firmy Novell.

JUNISOFTEX – to dostawca sprawdzonego sprzętu komputerowego renomowanych firm ALR, IBM, TEAM, DIGILAB, EPSON, MANNESMANN TALLY.

JUNISOFTEX – to wyłączny i autoryzowany dystrybutor komputerów i terminali firmy DIGILAB na Śląsku.

JUNISOFTEX – to dostawca i wykonawca okablowania lokalnych sieci komputerowych ARCNET, ETHERNET na kablach zwykłych i światłowodowych oraz na dwużyłowych kablach telefonicznych w przypadku stosowania oprogramowania NetWare ACCESS SERVER firmy Novell.

JUNISOFTEX – to nauczyciel, który chętnie podzieli się swoją wiedzą na organizowanych kursach w swojej szkole informatycznej – w tym nauczy Cię: podstaw informatyki, obsługi sprzętu komputerowego, eksploatacji własnych systemów informatycznych, edytorów tekstu: CHiWRITER, WordPerfect, arkuszy kalkulacyjnych QPRO v. 3.0, LOTUS 1-2-3 itp.

JUNISOFTEX – to doradca w zakresie komputerowych metod organizacji i eksploatacji systemów komputerowych.

JUNISOFTEX – to strażnik postępu i nowoczesności w Twojej firmie.

JUNISOFTEX – to gwarancja niezawodności, rzetelności i terminowości.

JUNISOFTEX – to Twój doradca i partner, któremu możesz zaufać, który Cię nigdy nie zawiedzie.

JUNISOFTEX – to partner, który Cię wysłucha i zawsze pomoże podjąć dobrą decyzję.

Jeżeli chcesz

- ✭ zastosować najnowocześniejsze systemy komputerowe oraz lokalne sieci komputerowe,
- ✭ zreorganizować, usprawnić i szybko skomputeryzować swoje przedsiębiorstwo,
- ✭ lepiej wykorzystać już istniejące w Twoim przedsiębiorstwie komputery,
- ✭ wymienić swoje niefortunnie zakupione oprogramowanie,
- ✭ pozbyć się problemów eksploatacyjnych, upadających systemów, nie działających komputerów,
- ✭ przeszkolić załogę swojego przedsiębiorstwa,
- ✭ dobrze zainwestować swoje pieniądze i podnieść rangę swojego przedsiębiorstwa,
- ✭ w przyszłości mieć dostęp do krajowej i światowej sieci komputerowej,

TO ZGŁOŚ SIĘ DO NAS!

0/17/91

JERZY SURMA
Akademia Ekonomiczna
Instytut Informatyki
Wrocław

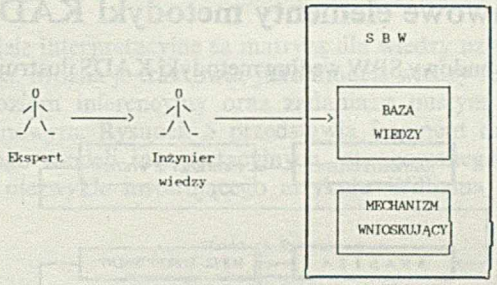
Modelowanie wiedzy w metodyce KADS

Metodyka tworzenia systemów z bazą wiedzy (SBW), a zwłaszcza problematyka modelowania wiedzy jest ciągle otwartym problemem badawczym. SBW, odznaczające się powodzeniem komercyjnym, są raczej efektem wirtuozerii ich twórców niż wynikiem podejścia inżynieryjnego. Wydaje się to być tym bardziej uzasadnione, że powszechny zachwyt nad udanymi realizacjami SBW na ogół zagłusza doniesienia o dziesiątkach upadłych projektów. Na dziś tworzenie SBW jest rzemiosłem, a nawet sztuką [20]. Niemniej w wielu ośrodkach uniwersyteckich są prowadzone badania nad formalizacją procesu budowy SBW. Do jednych z najbardziej obiecujących w tym zakresie osiągnięć należą prace Boba Wielingi i Joosta Breuker'a z Uniwersytetu w Amsterdamie, którzy w ramach programu ESPRIT pracują nad spójną i komercyjnie użyteczną metodyką budowy SBW, zwaną KADS (ang. *Knowledge Analysis and Documentation System*) [4, 5, 13, 21, 22]. W niniejszym artykule zostaną przedstawione w zarysie tylko najbardziej istotne aspekty metodyki KADS.

Należy podkreślić, że w podręcznikowych opisach metodyk budowy SBW [12, 17, 20] jakby nie zauważa się przydatności prac prowadzonych w zakresie modelowania konceptualnego baz danych oraz strukturalnej analizy systemów. W KADS korzysta się z osiągnięć obu tych dziedzin.

Akwizycja wiedzy

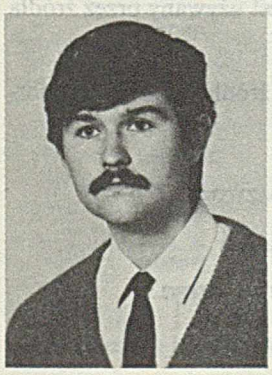
Modelowanie wiedzy jest częścią procesu pozyskiwania (akwizycji) wiedzy (ang. *knowledge acquisition*). Akwizycję wiedzy można zdefiniować jako proces transformacji wiedzy eksperta dotyczącej metod rozwiązywania danego problemu do bazy wiedzy SBW [12]. W klasycznym podejściu (rys. 1) akwizycja wiedzy jest prowadzona przez inżyniera wiedzy (ang. *knowledge engineer*).



Rys. 1. Rola inżyniera wiedzy w klasycznym procesie nabywania wiedzy

Generalnie można wyróżnić dwa podejścia do akwizycji wiedzy. Pierwsze, to tzw. szybka budowa prototypu (ang. *rapid prototyping*) oraz drugie, wykorzystujące metody podejścia poznawczego (ang. *cognitive science*). Szybka budowa prototypu polega na zapisaniu pierwszych reguł do bazy wiedzy już po pierwszych rozmowach z ekspertem. Następnie, przez ciągłe weryfikowanie systemu, następuje wprowadzanie do bazy wiedzy nowych reguł i modyfikowanie już istniejących [17]. Podstawowe wady tej metody, to rozmycie się podziału między metodą rozwiązywania problemu, a jej reprezentacją oraz brak jawnego modelu wiedzy, który utrudnia prace nad systemem. W opozycji do tej metody są techniki wypracowane w ramach podejścia poznawczego, gdzie wprowadzanie wiedzy do systemu jest poprzedzone dokładną analizą dziedziny, w tym szczególnie strategii rozwiązywania problemów używanych przez eksperta i zbudowaniem konceptualnego modelu wiedzy [14, 18]. Korzenie metodyki KADS można odnaleźć w drugim z omawianych powyżej podejść. Aby sprostać niezwykłej złożoności procesu konstruowania SBW, twórcy KADS proponują dekompozycję tego procesu przez tworzenie modeli odpowiadających różnym aspektom zadania stojącego przed inżynierem wiedzy. Są to modele: organizacyjny, aplikacji, zadania, konceptualny i konstrukcyjny. W dalszej części artykułu zostanie omówiony nieco szerzej model konceptualny.

Wiedza, która ma być reprezentowana w SBW pochodzi głównie od eksperta dziedzinowego i z literatury przedmiotu. Wiedza pochodząca od eksperta jest reprezentowana w języku naturalnym w postaci protokołu z wywiadu lub np. z nagrania „głośnego myślenia". Pojawia się tu problem, czy wiedza reprezentowana w mózgu eksperta, jak i opis problemu, dadzą się wyrazić werbalnie. Dla dalszych rozważań założono, że część wiedzy, która dała się wyrazić w języku naturalnym jest wystarczająca, aby problem można było rozwiązać na zadowalającym poziomie. Założenie to wydaje się być oczywiste, bo przecież jednym z zasadniczych form przekazu wiedzy jest



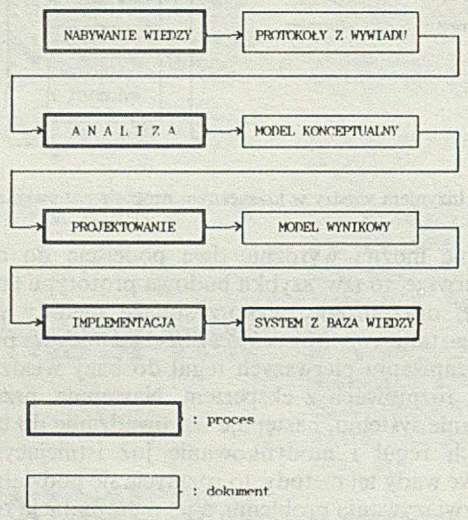
Mgr inż. JERZY SURMA ukończył w 1989 r. Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, kierunek Informatyka. Obecnie pracuje jako asystent w Instytucie Informatyki Akademii Ekonomicznej we Wrocławiu. Zajmuje się zagadnieniami inżynierii wiedzy, w tym szczególnie problematyką modelowania wiedzy oraz praktyczną realizacją systemów z bazą wiedzy.

język naturalny. Niemniej w sytuacji, gdy budowany system wymaga wiedzy nieformalnej eksperta, której na próżno szukać w podręcznikach, wydaje się, że udział np. zmysłu wzroku jest równie ważny, jak przekaz słowny.

Zasadniczym problemem jest takie przeprowadzenie analizy protokołu, aby wiedza, która w efekcie zostanie wprowadzona do SBW, była możliwie najwierniejszą kopią wiedzy eksperta. Może to być weryfikowane przez porównanie odpowiedzi systemu z odpowiedziami eksperta na tak samo postawione pytanie. Zasadniczym celem omawianej metodyki jest zbudowanie jawnego modelu wiedzy eksperta, modelu niezależnego od konkretnej implementacji systemu. Podstawą filozoficzną powyższych rozważań jest przekonanie co do prawdziwości hipotezy paradygmatu symbolicznego, która głosi, że ludzkie poznanie da się reprezentować za pomocą operacji na symbolach odpowiadającym różnym aspektom rzeczywistości [3, 7].

Podstawowe elementy metodyki KADS

Proces budowy SBW według metodyki KADS ilustruje rys. 2.



Rys. 2. Etapy budowy systemu z bazą wiedzy

W wyniku procesu nabywania wiedzy są otrzymywane tzw. protokoły z wywiadu oraz wypisy z literatury przedmiotu, które następnie podlegają analizie. Podczas analizy dokonuje się m.in. identyfikacji pojęć związanych z rozważaną dziedziną, rozpoznaje się powiązania interferencyjne, strategie rozwiązań. Głównym celem procesu analizy jest zbudowanie modelu konceptualnego opartego na czterech poziomach opisu wiedzy, które zostaną nieco dokładniej omówione w dalszej części artykułu. Etap ten można traktować jako odpowiednik procesu budowy schematu pojęciowego podczas modelowania przy budowie bazy danych. Etap ten jest często wspomagany przez użycie specjalnych edytorów protokołów. Przykładowo, na potrzeby KADS opracowano specjalny edytor specyfikacji [2, 16].

Po etapach projektowania oraz implementacji otrzymujemy SBW. Etapy te są związane z fizyczną realizacją SBW i nie będą przedmiotem niniejszego artykułu.

Model konceptualny wiedzy

Jak już wspomniano, w wyniku analizy wiedza eksperta jest formalizowana w postaci czteropoziomowego opisu, tj. poziomów: dziedzinowego, inferencyjnego, zadania i strategii. Jest to szczególny proces w metodyce KADS, który odróżnia konstruowanie SBW od tworzenia konwencjonalnych systemów informatycznych.

Poziom dziedzinowy. Wiedza na poziomie dziedzinowym ma głównie charakter statyczny i może być reprezentowana przez model danych sieci semantycznych. Model ten ma postać grafu, w którym wierzchołki reprezentują rzeczy, a krawędzie asercje o rzeczach. W sposób ogólny dziedzina jest opisywana przez:

- Obiekty (ang. *concepts*), które reprezentują rzeczy, pojęcia w danej dziedzinie. Obiekty są identyfikowane przez nazwę;
- Atrybuty: obiekty mają określone własności, charakterystyki zwane atrybutami. Atrybuty są definiowane przez nazwę i zbiory wartości dopuszczalnych;
- Związki między obiektami: obiekty można organizować w hierarchie według związków *IS_A, CONSIST_OF, DEPENDENT_UPON*, itp. Należy podkreślić, iż w hierarchii *IS_A*, każdy obiekt dziedziczy wszystkie atrybuty i ich wartości po obiekcie wyższego rzędu;
- Związki między atrybutami, a właściwie związki między wartościami atrybutów. Związki te mają zwykle charakter czasowy lub przyczynowy;
- Struktury, reprezentujące pojęcia złożone, np. obiekty złożone z innych obiektów.

W opisie wiedzy nie wspomina się o tym, jak korzystać z wiedzy poziomu dziedzinowego. Wiedza tego poziomu ma charakter aksjomatyczny, ponieważ w rzeczywistości zawiera fakty, na których opiera się podczas wnioskowania.

Poziom inferencyjny. Poziom ten reprezentuje powiązania inferencyjne w danej dziedzinie, bez specyfikowania sposobu korzystania z nich w trakcie rozwiązywania problemu. Można powiedzieć, że poziom inferencyjny zawiera metawiedzę o poziomie dziedzinowym. Metawiedza jest wyrażona w postaci tzw. źródeł, generatorów wiedzy (ang. *knowledge sources*). Argumentami źródeł wiedzy są metaklasy, które określają, jak pojęcia zdefiniowane na poziomie dziedzinowym będą używane podczas procesu wnioskowania. Przykładowo, w diagnostyce medycznej nazwa choroby, określona na poziomie dziedzinowym, będzie hipotezą na poziomie inferencyjnym. Źródła wiedzy reprezentują funkcjonalny opis procesu wnioskowania i mają co najmniej jeden parametr wejściowy, i co najmniej jeden parametr wyjściowy. Parametry te są metaklasami. Źródła wiedzy są wzajemnie uzależnione; przykładowo parametr wyjściowy w jednym źródle wiedzy może być parametrem wejściowym w innym źródle wiedzy. Powiązania inferencyjne mogą być reprezentowane w postaci grafu, gdzie etykietowane węzły będą źródłami wiedzy, a zorientowane i etykietowane krawędzie – argumentami we-wy tj. metaklasami. Reasumując, poziom inferencyjny jest określony przez źródła wiedzy, metaklasy i tzw. perspektywę dziedziny (ang. *domain view*), specyfikując sposób w jaki określony fragment dziedziny jest używany przez źródła wiedzy.

Na rys. 3 przedstawiono przykład źródła wiedzy. Oto formalna specyfikacja tego przykładu:

```
źródło_wiedzy: DEKOMPOZYCJA

    wejściowa_metaklasa:

        MODEL_SYSTEMU -> CZESC

    wyjściowa_metaklasa:
```

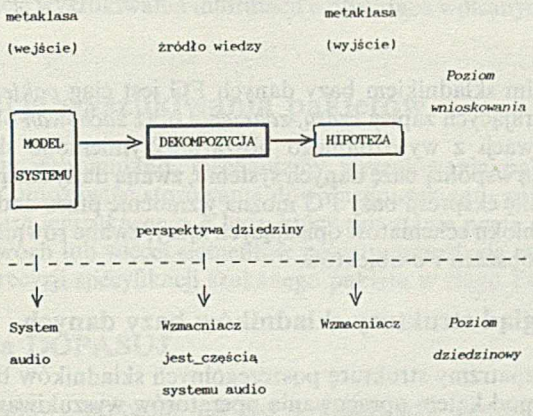
```
HIPOTEZA -> CZESC

widok_dziedziny:

    DEKOMPOZYCJA ( MODEL_SYSTEMU,HIPOTEZA ) ->

    jest_częścią ( CZESC,CZESC )
```

Strzałki określają, jak wiedza z poziomu inferencyjnego jest odwzorowywana w poziom dziedzinowy. W powyższym przykładzie perspektywa dziedziny odnosi się tylko do jednego typu wiedzy, tj. do relacji *jest_częścią*.

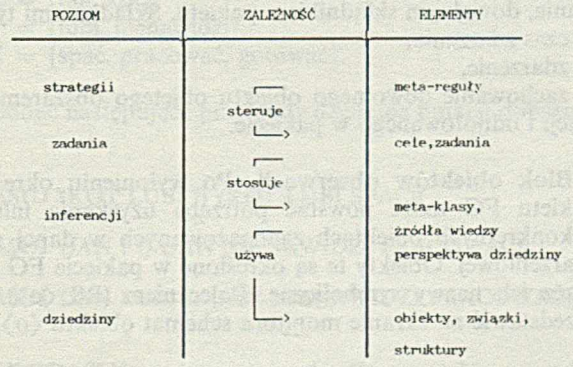


Rys. 3. Dekompozycja systemu audiowizualnego jako przykład źródła wiedzy

Poziom zadania. Na tym poziomie określa się cele, jakie będą osiągnięte w procesie rozwiązywania problemu. Pociąga to za sobą określenie metody osiągnięcia konkretnego celu, czyli sprecyzowania, jakie wnioskowania zostaną użyte oraz kiedy. Wiedzę na tym poziomie specyfikuje się przez określenie:

- Zadania (ang. *task*), które jest akcją wykonywaną w trakcie rozwiązywania problemu. Zastosowanie akcji do poszczególnych problemów prowadzi do realizacji celów;
- Oznaczeń sterujących (ang. *control terms*). Są to nazwy użyteczne dla elementów zbioru metaklas wykorzystywanych podczas sterowania procesem wnioskowania;
- Struktury zadania (ang. *task structure*) służącej do określenia dekompozycji na podzadania oraz zależności między podzadaniami. Jest to związane z dekompozycją celu na podcele, które jednocześnie reprezentują stany zaawansowania rozwiązania problemu. Zarówno cele, jak i podcele, są wybranymi metaklasami zdefiniowanymi na poziomie inferencyjnym.

Poziom strategii. Poziom strategii definiuje m.in. zachowania systemu w sytuacjach impasu, tj. poszukiwanie np. alternatywnych rozwiązań. Uwzględnia również aktualne możliwości dostępu do zewnętrznych źródeł wiedzy lub bieżące potrzeby użytkownika. Poziom strategii specyfikuje się za



Rys. 4. Poziomy opisu wiedzy eksperta

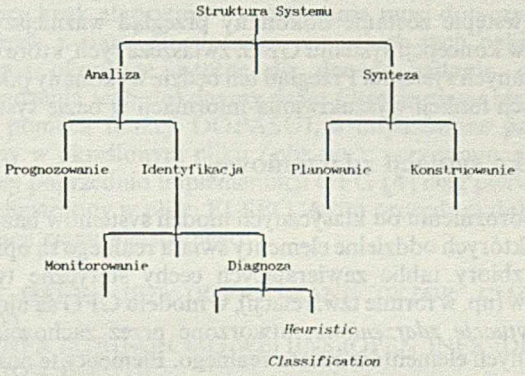
pomocą metareguł. Jest on obecnie najsłabiej opracowany i najczęściej pomijany w praktycznych zastosowaniach metodyki KADS.

Zależności między poszczególnymi poziomami opisu wiedzy ilustruje rys. 4. Model wiedzy eksperta, wyrażony w wyżej opisany sposób, jest niezależny od konkretnej implementacji.

Modele wielokrotnego użytku

Jednym z najbardziej oryginalnych aspektów omawianej metodyki jest wprowadzenie fragmentów modeli wiedzy wielokrotnego użytku (ang. *reusable model elements*). Badając wiele konkretnych aplikacji zauważono, że istnieją niektóre fragmenty modelu wiedzy, które się powtarzają i są niezależne od dziedziny. Na tej podstawie zaproponowano wspomaganie inżyniera wiedzy przez wprowadzenie biblioteki klasycznych modeli powtarzających się fragmentów wiedzy. Takie fragmenty wiedzy nazywa się modelami interpretacyjnymi (ang. *interpretation models*).

Modele interpretacyjne są matrycą dla wiedzy uzyskanej od eksperta. Można je traktować jako modele wiedzy reprezentujące poziom inferencyjny oraz zadania z pustym modelem dziedzinowym. Rysunek 5 przedstawia fragment drzewa decyzyjnego modeli interpretacyjnych, opracowanego na podstawie niezwykle inspirującego artykułu Williama Clancey'a [8].



Rys. 5. Fragment drzewa decyzyjnego modeli interpretacyjnych

Dla konkretnej aplikacji inżynier wiedzy wybiera model interpretacyjny z określonego węzła terminalowego drzewa decyzyjnego. Tego typu podejście jest bardzo wygodne z punktu widzenia inżyniera wiedzy. Niemniej w praktyce mogą występować znaczne rozbieżności między strategią rozwiązywania problemu przez eksperta, a strategią zaproponowanego modelu interpretacyjnego, co w efekcie może zakłócić proces akwizycji wiedzy.

Perspektywy metodyki KADS

W przeciwieństwie do wielu projektów badawczych, prace nad KADS doprowadziły do wymiernych zastosowań praktycznych. Obecnie na podstawie tej metodyki realizuje się ok. 40 projektów budowy SBW. O ogromnym zainteresowaniu świadczą zainicjowane w 1990 r. sympozja użytkowników KADS. Do najciekawszych zastosowań KADS należą systemy:

- konsultacji w zakresie wyboru metody statystycznej do analizy danych doświadczalnych [9],
- doradztwa ubezpieczeniowego [6],
- wspomagania leczenia chorób raka metodami chemioterapii [15].

dokończenie na s. 21

ZDZISŁAW WRZESZCZ
Warszawa

Operacje wyszukiwania w bazie danych systemu GFG

System GFG jest systemem umożliwiającym automatyczne gromadzenie informacji fotograficznej w bazie danych systemu oraz operowanie tymi danymi (wyszukiwanie i modyfikowanie). Zarówno gromadzenie, jak i operowanie informacją fotograficzną (*FG*) jest realizowane za pomocą środków sztucznej inteligencji (ang. *Artificial Intelligence* – AI).

W artykule poprzednim [4] omówiono pojęcie informacji faktograficznej, model systemu GFG oraz sposób implementacji funkcji gromadzenia. Obecny artykuł zawiera opis sposobu realizacji funkcji wyszukiwania w bazie danych systemu GFG.

Elementy koncepcji systemu GFG

Na wstępie zostanie dokonany przegląd ważniejszych elementów koncepcji systemu GFG, zwłaszcza tych, które dotyczą bazy danych systemu. Przegląd ten będzie wykonany pod kątem realizacji funkcji wyszukiwania informacji w bazie systemu.

Replika sytuacji zdarzeniowej

W odróżnieniu od klasycznych modeli systemów baz danych [2], w których oddzielne elementy świata realnego są opisywane przez zbiory tablic zawierających cechy statyczne tych elementów (np. w formie tzw. relacji), w modelu GFG są ujmowane całe *sytuacje zdarzeniowe*, utworzone przez zachowanie wyróżnionych elementów świata realnego. Elementy te nazywamy *obiektami obserwacji* (o-obiektami).

Mechanizmy systemu GFG umożliwiają utworzenie w pamięci systemu struktur danych, reprezentujących w spójny sposób (w formie *schematów*) wyróżnione o-obiekty wraz z ich statycznymi i *dynamicznymi* własnościami. Mówimy wtedy, że w pamięci systemu została utworzona *replika rzeczywistej sytuacji*, w której następują interesujące nas *zdarzenia*. Zdarzenia te zachodzą w związku z zachowaniem obiektów aktywnych (obiekty roli *A*) i dzięki sprzyjającemu zachowaniu obiektów wspierających (obiekty roli *S*); zdarzenia są manifestowane przez zachowanie obiektu pasywnego (obiekt roli *P*). Wszystkie wymienione obiekty obserwacji należą do tego samego obszaru obserwacji.

Baza danych systemu

W związku z utworzeniem repliki sytuacji problemowej, w systemie GFG powstaje baza danych zawierająca informację o obszarze obserwacji i o obiektach wszystkich trzech ról (*A, S, P*). Struktura tych informacji jest jednorodna i ma postać schematu będącego zbiorem pól wypełnianych nazwą atrybutu obiektu i wartością tego atrybutu. Pola schematów są wypełniane danymi dotyczącymi konkretnych obiektów tworzących daną sytuację *zdarzeniową*. Wypełnianie to następuje w wyniku dialogowym *procesów inicjacji* bazy danych, dotyczących obszaru obserwacji oraz zawartych w tym obszarze obiektów. Trzecim składnikiem bazy danych FG jest ciąg *pakietów* FG zawierających zapisy *czasu, zdarzenia* oraz *zachowań* obiektów obserwacji z wyróżnionego obszaru. Wymienione składniki tworzą wspólną bazę danych systemu, zwaną dalej bazą danych FG. Siłę ekspresji bazy FG można wzmocnić przez dodanie do bazy bloku schematów opisujących *klasy*, zwane również *wzorcami* obszaru i o-obiektów.

Przegląd struktury składników bazy danych

Rozpatrzmy strukturę poszczególnych składników bazy danych pod kątem opracowania operatorów wyszukiwania.

Blok ciągu FG. Blok ten zawiera syntetyczne opisy obserwowanych sytuacji zdarzeniowych. Przypomnijmy sobie zapis ciągu FG:

⟨ciąg FG⟩ := ⟨pakiet FG⟩ ⟨pakiet FG⟩ ... ⟨pakiet FG⟩

Jest to więc ciąg pakietów FG o następującej postaci:

```
⟨pakiet FG⟩ := ((⟨m⟩ ⟨d⟩ ⟨h⟩ ⟨min⟩)   ; czas zdarzenia
                (⟨po⟩ ⟨bat-n⟩ ⟨bat-v⟩) ; zdarzenie
                (⟨ao⟩ ⟨bat-n⟩ ⟨bat-v⟩) ; zachowanie obiektu
                                          aktywnego
                (⟨so⟩ ⟨bat-n⟩ ⟨bat-v⟩) ; zachowanie obiektu
                                          wspierającego
                ..........................  ..................
                (⟨so⟩ ⟨bat-n⟩ ⟨bat-v⟩)) ; zachowanie obiektu
                                          wspierającego
```

gdzie:
m – miesiąc, d – dzień, h – godzina, min – minuta, po – symbol obiektu pasywnego, ao – symbol obiektu aktywnego, so – symbol obiektu wspierającego, bat-n – nazwa atrybutu zachowania, bat-v – wartość atrybutu zachowania; jeśli przyjmiemy, że x jest dowolnym z powyższych oznaczeń, to ⟨x⟩ jest dowolnym symbolem z ustalonego zbioru X.

Jest wysoce pożądane, by w systemie istniała możliwość wyszukiwania pakietów posługując się, jako kluczem wyszukiwania, dowolnym składnikiem pakietu. Składnikami tymi są:
- czas zdarzenia,
- zdarzenie,
- zachowanie dowolnego obiektu objętego obszarem obserwacji i odnotowanego w pakiecie.

Blok obiektów obserwacji. Po wyłonieniu określonego pakietu FG może powstać potrzeba uzyskania informacji o konkretnych obiektach zaangażowanych w danej sytuacji zdarzeniowej. Obiekty te są określone w pakiecie FG jedynie przez ich nazwy symboliczne. Poleceniem (RS ⟨o⟩) można przedstawić na ekranie monitora schemat obiektu ⟨o⟩, np.:

```
>>(RS '03)
(06 (NAZWA (VA RUCH)
```

```
(ROLA (VA A))
(KLASA (VA DRUŻYNA-PIŁKARSKA))
(BAT (VA AKT-SPORT (BRAK SLABA NORMALNA SILNA)))
(POZYCJA (VA 6))
(...)))
```

pod warunkiem, że 03 znajduje się w pamięci operacyjnej systemu. W przeciwnym przypadku 03 musi być uprzednio przywołane do pamięci poleceniem (WYW-OBIEKTY).

W rezultacie są potrzebne dwie grupy operacji wyszukiwania:
- operacje wyszukiwania pakietów,
- operacje wyszukiwania informacji o obiektach wpisanych do pakietu.

Operacje wyszukiwania pakietów

Jak już wspomniano, mechanizmy wyszukiwania pakietu powinny umożliwiać stosowanie dowolnego elementu pakietu jako *klucza wyszukiwania*. Klucz, który powstanie przez połączenie dwóch lub więcej elementów pakietu, umożliwia zwiększanie precyzji specyfikacji szukanego pakietu w ciągu FG.

Funkcja DOPASUJ

Do budowy procedur takiego wyszukiwania zastosujemy metodykę nazywaną *dopasowanie wzoru* (ang. *pattern matching*). Powstanie tej metodyki wiąże się z wczesnymi badaniami nad maszynowym przetwarzaniem wyrażeń języka naturalnego. U podstaw tych prac leżało przekonanie, że informacja syntaktyczna zawarta w danym wyrażeniu, połączona ze znaczeniem słów należących do zbioru skończonego jest dostateczna do realizacji eksperymentów, takich jak prowadzenie dialogu z komputerem na dany temat. W drugiej połowie lat sześćdziesiątych powstało wiele programów, takich jak SIR [3], służących do przetwarzania wyrażeń języka naturalnego. I chociaż uzyskiwane wyniki tylko w części zadowalały, to jednak zastosowana tu metodyka dopasowania wzorów znalazła odtąd stałe miejsce w repertuarze środków AI.

Podstawowym członem wymienionych programów (pisanych przeważnie w Lispie) była funkcja dopasowująca podawane wyrażenia do bazy wzorów typu:

```
⟨x⟩ ? ... ?
⟨x⟩ ⟨y⟩ ... ?
? ? ... ⟨z⟩
.....
```

Zbiór tych wzorów jest ograniczony, tak samo jak zbiory wyrazów X, Y, Z zawierające symbole ⟨x⟩, ⟨y⟩, ⟨z⟩. Symbol ? oznacza dowolny wyraz ze zbiorów X, Y, Z.

```
Niech: X = {jan, kot},
       Y = {lubi, nienawidzi},
       Z = {spać, pracować, gotować}.
```

Można podać następujące przykłady działania funkcji dopasowania:

```
>>(DOPASUJ '(KOT LUBI ?) '(KOT LUBI SPAĆ))
T
>>(DOPASUJ '(JAN ? ?) '(JAN NIENAWIDZI PRACOWAĆ))
T
>>(DOPASUJ '(JAN LUBI ?) '(JAN NIENAWIDZI PRACOWAĆ))
NIL
```

Funkcja DOPASUJ, zapisana w języku Common Lisp, przedstawia się następująco:

```
(DEFUN DOPASUJ (WZOR1 WZOR2)
(COND ((OR (ATOM WZOR1) (ATOM WZOR2)) (DOPAS-ATOM
      WZOR1 WZOR2))
      (T (AND (DOPASUJ (CAR WZOR1) (CAR WZOR2))
              (DOPASUJ (CDR WZOR1) (CDR WZOR2))))))
```

WZOR1 jest wzorem, do którego są dopasowywane wyrażenia podstawiane jako WZOR2. Mogą to być elementy bazy danych. Konstrukcja DOPASUJ zakłada, że dwa s-wyrażenia (WZOR1, WZOR2) pasują, jeśli tylko ich wartości CAR i CDR pasują odpowiednio. Funkcja ma postać rekursji, a bieg rekursji zostaje przerwany wtedy, gdy dowolny z argumentów jest atomem (lista pusta lub symbol NIL). Wynikiem DOPASUJ jest T, jeśli WZOR1 i WZOR2 są tym samym atomem lub gdy jeden wzór jest zmienną atomową „?". W innym przypadku wynikiem jest NIL, czyli brak dopasowania. Funkcja DOPASUJ ma klasyczną budowę lispową – wykorzystuje podstawowe mechanizmy tego języka.

Funkcje pomocnicze dla funkcji DOPASUJ są sformułowane następująco:

```
(DEFUN DOPAS-ATOM (WZOR1 WZOR2)
(OR (EQUAL WZOR1 WZOR2) (ZMIENNA WZOR1) (ZMIENNA
WZOR2)))
(DEFUN ZMIENNA (X) (EQUAL X '?))
```

Procedura DAJ-PAKIETY-DOPASOWANE

Pierwszy krok algorytmu wyszukiwania musi dotyczyć wyprowadzenia z ciągu pierwotnego wszystkich tych pakietów, które mają określone, istotne dla danego zadania, cechy. Wyprowadzenia tego dokonamy przez filtrację ciągu pierwotnego za pomocą funkcji DOPASUJ, a dopasowane pakiety umieścimy w określonym pliku ⟨plik-wy⟩; przypominamy, że w opisanej poprzednio implementacji GFG [4] ciąg pierwotny został umieszczony w pliku FI-SEQ. A oto procedura dokonująca filtracji ciągu pierwotnego:

```
(DEFUN DAJ-PAKIETY-DOPASOWANE (WZOR CIAG-WE
CIAG-WY)
(DO * ((INFX (STRING CIAG-WE) (INS (OPEN INFX :DIREC-
       TION :INPUT))
       (PAK (READ INS NIL 'NIC)) (OUTFX (STRING
       PLIK-WY))
       (OUTS (OPEN OUTFX :DIRECTION :OUTPUT)))
((EQUAL PAK 'NIC) (CLOSE INS) (CLOSE OUTS) (TERPRI)
(PRINC "@
DOPASOWANE PAKIETY ZNAJDUJA SIE W PLIKU ") (PRIN1
CIAG-WY))
(COND ((DOPASUJ WZOR PAK) (PRINT PAK OUTS)))
(SETQ PAK (READ INS NIL 'NIC))))
```

Pakiety są pobierane z pliku pierwotnego ⟨CIAG-WE⟩ przez strumień wejściowy INS; pakiet bieżący PAK, po dopasowaniu do wzoru WZOR jest kierowany do pliku wyjściowego ⟨CIAG-WY⟩ za pośrednictwem strumienia wyjściowego OUTS. Dopasowanie bada oczywiście funkcja DOPASUJ, po czym następuje pobranie kolejnego pakietu z ciągu pierwotnego. Po każdym pobraniu jest badana zawartość strumienia wejściowego: jeśli strumień INS nie zawiera nic, tzn. PAK jest równy NIC, to iteracja zostaje zatrzymana i następuje przekazanie komunikatu „DOPASOWANE PAKIETY ZNAJDUJA SIE W PLIKU ⟨CIAG-WY⟩, a pod symbol CIAG-WY zostaje podstawiona konkretna nazwa pliku dyskowego.

Prześledźmy działanie procedury na przykładzie następującego ciągu pierwotnego:

```
>>(FLPRIN 'PLIK1
((9 1 8 0) (01 REZULTAT WYGRANA) (02 POGODA DOBRA)
```

((9 2 8 0) (01 REZULTAT PRZEGRANA) (02 POGODA ZLA))
((9 2 15 0) (01 REZULTAT WYGRANA) (02 POGODA ZLA))

Powyższy ciąg pakietów FG opisuje trzy sytuacje składające się z zapisu czasu, zapisu zdarzenia (obiekt 01) oraz zachowania obiektu 02 określającego warunki atmosferyczne. Niech wzór WZ ma następującą wartość:

```
>>(SETQ WZ ' (9 1 8 0) ? ?))
((9 1 8 0) ? ?)
```

Dokonując filtracji podanego wyżej ciągu pierwotnego

```
>>(DAJ-PAKIETY-DOPASOWANE WZ 'PLIK1 'PLIK2)
@DOPASOWANE PAKIETY ZNAJDUJA SIE W PLIKU PLIK2
PLIK2
```

uzyskamy w pliku PLIK2:

```
>>(FLPRIN 'PLIK2)
((9 1 8 0) (01 REZULTAT WYGRANA) (02 POGODA DOBRA))
```

Wzór WZ w pełni określał jeden ze składników, natomiast pozostałe składniki były opisane jako zmienne. Jest możliwe również wprowadzenie do środka składnika znaku zmiennej. Niech WZ ma wartość:

```
>>(SETQ WZ '((9 ? ? 0) (01 REZULTAT PRZEGRANA) (02 POGODA
?)))
((9 ? ? 0) (01 REZULTAT PRZEGRANA) (02 POGODA ?))
```

Po filtracji ciągu pierwotnego uzyskamy:

```
>>(DAJ-PAKIETY-DOPASOWANE WZ 'PLIK1 'PLIK2)
@DOPASOWANE PAKIETY ZNAJDUJA SIE W PLIKU PLIK2
PLIK2
>>(FLPRIN 'PLIK2)
((9 2 8 0) (01 REZULTAT PRZEGRANA) (02 POGODA ZLA))
NIL
```

Operacje na obiektach obserwacji

Po wyłonieniu określonego pakietu, np. tak jak w przypadku procedury DAJ-PAKIETY-DOPASOWANE, może być potrzebne zapoznanie się z obiektami, występującymi w tym pakiecie. O-obiekty występują w systemie pod następującymi dwiema postaciami:

- jako plik atrybutów zapisany w pamięci dyskowej pod nazwą odpowiadającą symbolowi o-obiektu,
- jako schemat obiektu, utworzony w pamięci operacyjnej przy inicjacji obiektu (operacja INI-OBIEKTY), lub też po wywołaniu bazy o-obiektów (operacja WYW-OBIEKTY).

Dla programisty bardziej czytelny jest schemat O-obiektu, gdy dowolny schemat ⟨sch⟩ zawarty w pamięci systemu może być przedstawiony na ekranie monitora w wyniku operacji (RS ⟨sch⟩). Na przykład:

```
>>(RS '06)
(06 (NAZWA (VA FABRYKA-AKUMULATOROW-NO-22))
    (KLASA (VA CO3))
    (ROLA (VA A))
    (BAT (VA AKT-PROD (BRAK SLABA NORMALNA SIL-
    NA)))
```

Widzimy, że obiekt o symbolu 06 jest klasy CO3. W systemie zachowuje się on tak jak obiekt aktywny (czyli powodujący zajścia zdarzenia), zaś jako atrybut zachowania tego obiektu przyjęto jego aktywność produkcyjną (AKT-PROD). Dziedzina tego atrybutu zawiera cztery wartości. Więcej danych o obiekcie 06 można uzyskać po wywołaniu schematu klasy CO3.

Manipulacja atrybutami o-obiektu (tj. klatkami schematu) jest możliwa za pomocą następujących funkcji:

- WPROWI – wprowadzanie atrybutu,
- WEZI – pobranie wartości wskazanego atrybutu,
- USUNI – usunięcie wartości wskazanego atrybutu.

Tak więc system dysponuje środkami umożliwiającymi podejmowanie decyzji sterujących procesem wyszukiwania. Decyzja taka może być podjęta zarówno przez programistę poza programem, jak i automatycznie przez sam system.

Dialogowe procesy wyszukiwania

Przedstawiony zespół operacji wyszukiwania umożliwia dostęp do dowolnej danej FG zawartej w bazie. Operacje te, wraz z operacjami demonstrującymi schematy i pliki, a także w połączeniu z operacjami gromadzenia [4], mogą być traktowane jako rozszerzenie środowiska Commpn Lisp (nazwijmy takie środowisko skrótem GFG-Lisp). Pisanie programów w GFG-Lisp będzie łatwe dla programistów Lispu, chociaż innego zdania mogą być wszyscy inni użytkownicy systemu. W związku z tym wydaje się niezbędne wprowadzenie dodatkowych środków ułatwiających użytkownikowi systemu realizację czynności wyszukiwania. Środkiem takim może być problemowo zorientowany język do gospodarki zasobami bazy danych z grupy języków określanych nazwą *Database Query Languages* np. język SQ1 (Structured Query Language) [1]. Ten syntaktycznie prosty język ma wiele konstrukcji umożliwiających określenie warunków wyszukiwania w bazie. Na przykład wyrażenie:

```
SELECT placa
FROM kierownictwo
WHERE placa > 4 000 000
```

formułuje wybór wszystkich pól „płaca" z pliku „kierownictwo", które spełniają warunek „WHERE placa > 4 000 000". Przyglądając się bliżej konstrukcjom SQL można dostrzec dziedzinową orientację tych konstrukcji. Szczególnie dotyczy to selektorów warunku, gdzie dominują operacje na danych liczbowych, typowe dla dziedziny aplikacji SQL. Do dziedziny tej należą różnego rodzaju zadania handlowe, zadania dotyczące zarządzania, księgowości itp.

W systemie GFG rezygnujemy chwilowo z formułowania języka typu *Database Query Language*, zastosujemy natomiast w to miejsce pewien zbiór *dialogowych procesów wyszukiwania* z wewnętrznymi środkami selekcji i struktur danych. Procesy te, podobnie jak dialogowe procesy gromadzenia [4], zawierają całą wiedzę dotyczącą sposobu selekcji zwalniając użytkownika od obowiązku pamiętania, zwłaszcza reguł syntaktycznych. Jako przykład przedstawimy jeden z tych procesów, kładąc nacisk na jego formę zewnętrzną, widzianą przez użytkownika systemu.

Dialogowy proces wyszukiwania pakietów

Przebiegiem procesu kieruje wiedza o strukturze pakietu, odwzorowana w procedurze procesu. Głównym składnikiem operacyjnym procedury procesu jest procedura DAJ-PAKIETY-DOPASOWANE, a rolę wiodącą w dialogu pełni system. Proces wyszukiwania przebiega następująco:

```
>>(PD-WYBIERZ-PAKIETY ⟨plik-zrodla⟩ ⟨plik-wyniku⟩)
@PODAJ CECHY WYBORU,
$ POTWIERDZ CECHE CZASU [+] [?]:
$ PODAJ WARTOSC ⟨M⟩ (1 .. 12):
$ PODAJ WARTOSC ⟨D⟩ [1 .. 31] [?]:
$ PODAJ WARTOSC ⟨H⟩ [1..24] [?]:
```

```
$ PODAJ WARTOSC ⟨MIN⟩ [12..60] [?]:
$ POTWIERDZ ZACHOWANIE OBIEKTU
  JAKO CECHE WYSZUKIWANIA [+] [?]:
$$ PODAJ SYMBOL OBIEKTU [⟨o⟩] [?]:
$$ PODAJ NAZWE B-ATRYBUTU [⟨bat-n⟩] [?]:
$$ PODAJ WARTOSC B-ATRYBUTU [⟨bat-v⟩] [?]:
.....
$ POTWIERDZ ZACHOWANIE OBIEKTU
  JAKO CECHE WYSZUKIWANIA [+] [?]:
$$ PODAJ SYMBOL OBIEKTU [⟨0⟩] [?]:
$$ PODAJ NAZWE B-ATRYBUTU [⟨bat-n⟩] [?]:
$$ PODAJ WARTOSC B-ATRYBUTU [⟨bat-v⟩] [?]:
@ZADANE PAKIETY ZNAJDUJA SIE W PLIKU ⟨plik-wy⟩
```

Obecna, druga część charakterystyki systemu GFG została poświęcona operacjom wyszukiwania danych w bazie FG. Jako podstawę realizacji tych operacji wybrano metodę AI, nazywaną „dopasowaniem wzoru", co pozwoliło opracować oszczędne i funkcjonalnie przejrzyste środki wyboru pakietów FG według dowolnej cechy strukturalnej pakietu.

Po opracowaniu środków do inicjacji systemu i gromadzenia danych FG (zbiór procedur zrealizowanych w pierwszej części charakterystyki systemu [4]) oraz środków do wyszukiwania danych FG w bazie systemu, dysponujemy podstawowymi narzędziami do tworzenia i eksploatacji bazy informacji faktograficznej. Określone tu środowisko GFG-Lisp pozwala formułować i realizować w prosty sposób dowolne zadania, polegające na dokumentowaniu sytuacji zdarzeniowych oraz na badaniu związków przyczynowo-skutkowych między elementami tych sytuacji. Badając takie związki można korzystać z metod pochodzących z rzeczywistych sytuacji, oczywiście po ich uprzednim odwzorowaniu w systemie.

LITERATURA

[1] Chamberlain D. D. i inni: SEQUEL 2: a unified approach to data definition, manipulation and control. IBM J. of Res. and Dev., vol. 20, No. 6, 1976

[2] Date C. J.: An Introduction to Database Systems, vol. 1. Addison-Wesley, Reading Ma., 1990

[3] Raphael B.: SIR: A computer program for semantic information retrieval. W zbiorze: M. Minsky (ed.): Semantic information processing. MIT Press, Cambridge Ma., 1968

[4] Wrzeszcz Z.: Inteligentny system gromadzenia informacji faktograficznej. INFORMATYKA, nr 6, 1992.

Wskazówki dla Autorów

Nadsyłane artykuły nie mogą być publikowane lub przeznaczone do opublikowania w innych czasopismach.

Materiał oprócz tekstu zasadniczego powinien zawierać:

- ★ krótki życiorys zawodowy Autora i jego zdjęcie,
- ★ wykaz literatury,
- ★ tabele,
- ★ materiał ilustracyjny (rysunki, zdjęcia czarno-białe, wydruki) dołączony do artykułu (nie wklejać materiału w tekst),
- ★ podpisy pod ilustracje.

Na osobnej stronie prosimy podać: tytuł naukowy, nazwisko i imię, nazwę zakładu pracy, numer telefonu służbowego oraz informację, jaką drogą przekazać honorarium – kasa Wydawnictwa, poczta, konto.

W związku z naliczaniem podatku dochodowego prosimy także podać:

- ★ oprócz nazwiska i imienia – także drugie imię,
- ★ imiona rodziców,
- ★ datę i miejce urodzenia,
- ★ numer identyfikacyjny PESEL,
- ★ miejsce zameldowania,
- ★ adres Urzędu Skarbowego właściwego dla miejsca zamieszkania Autora.

Tekst artykułu prosimy dostarczyć w formie maszynopisu lub wydruku komputerowego, pisanego jednostronnie, z podwójnym odstępem (30 wierszy na stronie i 60 znaków w wierszu).

Rysunki powinny być czytelne i mogą być wykonane ołówkiem.

Autor opublikowanego artykułu otrzymuje honorarium i bezpłatny egzemplarz okazowy.

Materiałów nie zamówionych redakcja nie zwraca.

Modelowanie wiedzy w metodyce KADS

dokończenie ze str. 17

Sukcesy rynkowe stymulują prace teoretyczne nad KADS w wielu ośrodkach uniwersyteckich. Do interesujących należą badania prowadzone na Wolnym Uniwersytecie w Brukseli, np. analiza zastosowań niezależnych dziedzinowo modeli metod rozwiązywania problemów [10], czy prace nad opracowaniem języka modelowania wiedzy [1, 11], realizowane na Uniwersytecie w Karsruhe. Pod kierunkiem twórców KADS obecnie są realizowane prace z zakresu budowy oprogramowania wspomagającego inżyniera wiedzy (system Shelley [2]) oraz prace nad formalnym językiem opisu wiedzy (ML^2 [19]). Na dzień dzisiejszy omawiana metodyka jest rozwijana w ramach programu ESPRIT pod nazwą KADS-2 z celem zrealizowania rzeczywistego standardu kompletnej metodyki budowy SBW.

LITERATURA

[1] Angele J., Fensel D., Landes D., Studer R.: Modelling an Assignment Problem with KARL. Bericht 216, Universität Karlsruhe, 1991

[2] Anjewierden A., Wielemaker J., Toussaint C.: Shelley – computer aided knowledge engineering. Knowledge Acquisition 4(1), 1992

[3] Bielecki M., Zytkow J.: Sztuczna inteligencja. Zdrowy rozsądek i filozofia. Studia filozoficzne 12(289), PWN 1989

[4] Breuker J., Wielinga B.: Use of Models in the Interpretation of Verbal Data. In: ed. Kidd A.: Knowledge Acquisition for Expert Systems. A practical handbook. Plenum Press 1987

[5] Breuker J., Wielinga B.: Models of Expertise in Knowledge Acquisition. In: ed. Guida G., Tasso C.: Topics in Expert System Design. Methodology and Tools. North Holland 1989

[6] Brunet E., Toussaint C.: A KADS application in insurance. ESPRIT P1098, Cap. Sesa Innoation 1990

[7] Compton P., Jansen R.: A philosophical basis for knowledge acquisition. Knowledge Acquisition 2(3) 1990

[8] Clancey W.: Heuristic Classification. Artificial Intelligence 27(3), 1985

[9] De Greef P., Breuker J., Schreiber G., Wielemaker J.: StatCons: Knowledge acquisition in a complex domain. In Proceedings ECAI'88, Munich 1988

[10] Duursma C.: A task independent description of role limiting methods. ESPRIT Project P5248 KADS-2, VUB AI Lab. 1992

[11] Fensel D.: An Introduction to KADS and KARL. Bericht 226, Universität Karlsruhe (TH) 1991

[12] Hayes-Roth F., Waterman D., Lenat D.: Building Expert Systems. Addison Wesley 1983

[13] Hayward S., Wielinga B., Breuker J.: Structured analysis of knowledge. In: ed. Boose J., Gaines B.: Knowledge Acquisition Tools for Expert Systems. Academic Press 1988

[14] Klein M., Methlie L.: Expert Systems. A Decision Support Approach. Addison Wesley 1990

[15] Linster M., Musen M.: Use of KADS to create a conceptual model of the ONCOCIN task. Knowledge Acquisition 4(1), 1992

[16] Macintosh A.: Using Shelley to produce a design specification. AIRING – News & Information from AIAI Nr. 11, 1990

[17] Pedersen K.: Expert Systems Programming. A practical techniques for rule-based systems. John Willey 1989

[18] Sowa J.: Conceptual Structures – Information Processing in Mind and Machine. Addison Wesley 1984

[19] Van Hermelen F., Balder J.: (ML^2): A formal language for KADS models of expertise. Knowledge Acquisition 4(1), 1992

[20] Walters J., Nielsen N.: Crafting Knowledge-Based Systems. Expert Systems made realistic. John Willey 1988

[21] Wielinga B., Breuker J.: Interpretation of verbal data for knowledge acquisition. In: ed. Oshea T.: Advances in AI, Proceed. ECAI 84. North Holland 1985

[22] Wielinga B., Shreiber A., Breuker J.: KADS: a modelling approach to knowledge engineering. Knowledge Acquisition 4(1), 1992.

AGNIESZKA SZEWCZYK
Instytut Cybernetyki Ekonomicznej
i Informatyki
Uniwersytet Szczeciński

Diagnostyka systemów informacyjnych (2)

Model diagnostyki systemu informacyjnego

Opis systemu informacyjnego przy wykorzystaniu faset substratów diagnostycznych ułatwia identyfikację tych obszarów i elementów systemu, które stanowią potencjalne źródła danych, umożliwiających procesy diagnozowania tego systemu.

Problemy wyboru, wartościowania i badania substratów diagnostycznych

Zgodnie z przyjętymi założeniami, format języka opisu systemu informacyjnego powinien być na tyle uniwersalny, aby umożliwiał pełny zakres diagnozowania systemów informacyjnych różnych klas. Spełnienie warunku niezbędnej różnorodności formatu języka opisu przedmiotu badań sprawia, że podstawowym problemem w procesie diagnozowania pozostaje dobór adekwatnych do klasy danego systemu substratów diagnostycznych.

Punktem wyjścia do rozwiązania tego problemu jest analiza procesu wartościowania. Badanie tego procesu jest jednak wciąż jednym z trudniejszych zagadnień empirycznych. Psychologowie nie stworzyli dotąd dostatecznie realnych i trafnych metod, za pomocą których można by przewidzieć i wyjaśnić przebieg procesu wartościowania czy też określania użyteczności. Szczególnie trudno jest badać użyteczność elementów mało wymiernych. Dodatkowym problemem jest ponadto fakt, że prawa rządzące procesami wartościowania, czy oceniania użyteczności ulegają modyfikacji pod względem zmiany celów decydenta.

W przypadku badań profilaktycznych systemu informacyjnego jest celowe wykorzystanie możliwie szerokiego zestawu substratów z różnych faset tematycznych. Jeżeli natomiast podjęcie działań diagnostycznych jest konsekwencją pojawienia się i zaobserwowania charakterystycznych, nietypowych, nieprawidłowych objawów w budowie lub w funkcjonowaniu systemu informacyjnego, to celowe jest wtedy rozpoczęcie działań od substratów mających związek z tymi objawami.

Kolejnym krokiem w procesie wyboru substratów diagnostycznych będzie ustalenie, czy zachodzi możliwość wykluczenia z badań pewnego obszaru diagnozowania, zakreślonego tematycznym ujęciem fasetowym. Jest to związane przede wszystkim z wyborem między substratami o charakterze jakościowym, ilościowym bądź kosztowym. Wybór ten powinien być uwarunkowany również możliwością zbadania, pomiaru oraz obliczeń wartości badanych cech systemu w danych warunkach [2].

Przy rozwiązywaniu problemu doboru substratów diagnostycznych należy dalej określić, głównie w zależności od miejsca i typu obserwowanych objawów, jakie substraty będą preferowane, biorąc pod uwagę ich charakter: sytuacyjny, osiągnięć, działalności lub struktury.

Na ograniczenie wyboru substratów znaczący wpływ ma typ diagnozowanego systemu informacyjnego. Pewne substraty są bowiem nieistotne dla określonych klas systemów. Chodzi tu zwłaszcza o rozróżnienie typu systemu informatycznego, którego diagnozowanie wymaga szeregu specyficznych substratów, zbędnych w warunkach systemów tradycyjnych.

Przy wybieraniu konkretnych substratów diagnostycznych jest pomocna analiza ich typów, podanych w ramach fasety, a także powiązań wewnętrznych i zewnętrznych oraz ich charakteru i długości. Można bowiem przystąpić do diagnozowania systemu wybierając pojedynczy substrat, uznany w danym przypadku za najistotniejszy, i kolejno przechodzić do następnych drogą łańcucha powiązań wewnętrznych i zewnętrznych. Celowe wydaje się zwrócenie uwagi na możliwość stopniowego wybierania substratów i poszerzania ich repertuaru wtedy, gdy wystąpi konieczność przeprowadzania dodatkowych badań.

Oprócz przedstawionych ogólnych zasad postępowania przy doborze substratów diagnostycznych, istotne wydają się następujące uwarunkowania, zgodnie z którymi pasywacja substratów może wynikać z:

- konieczności ograniczenia kosztów diagnostyki,
- konieczności skrócenia czasu diagnozowania,
- ograniczonych możliwości zespołu diagnozującego,
- możliwości uzyskania tylko niepełnych danych lub ich całkowitego braku w odniesieniu do konkretnych substratów. Może wynikać to z faktu np. braku dostępnych metod i narzędzi pomiaru wartości niektórych substratów lub z niemożności posługiwania się określonymi metodami badań, a także z niechętnej postawy użytkowników systemu, którzy odmawiają współpracy w trakcie badań.

Jak wynika z przeprowadzonych sondaży wypowiedzi użytkowników, często mogą one jedynie sugerować zakres i głębokość badania systemu ze względu na zazwyczaj ogólną postać typu: coś jest nie tak w obiegu dokumentów, w sprzęcie, w programie, w atmosferze biura, ale nie bardzo wiadomo, co i dlaczego.

Wybranie lub pominięcie określonych substratów diagnostycznych wiąże się w prezentowanej metodzie diagnozowania z nadaniem im charakteru aktywnego, bądź pasywnego. Proponuje się przypisanie substratom, w zależności od dokonanego

wyboru, następujących wskaźników:
- substrat aktywny, uwzględniany w procesach diagnostycznych danego systemu – A,
- substrat pasywny, nieuwzględniany w procesach diagnostycznych danego systemu – P.

Kolejnym krokiem w procesie wyboru i wartościowania substratów diagnostycznych będzie określenie stopnia istotności substratów aktywnych dla celów diagnozowania systemu, czyli tego, czy dany substrat jest typowy czy nietypowy oraz konieczny lub niekonieczny dla realizacji procedury diagnozowania.

Wartościowania tych substratów można dokonać przy uwzględnieniu:
- częstotliwości występowania substratu, w przypadku gdy reprezentuje on dany symptom,
- stopnia negatywnego wpływu danego substratu na stan systemu,
- możliwości odnowy systemu po wyeliminowaniu danego substratu.

Uzyskany zestaw aktywnych substratów diagnostycznych wraz z określeniem ich mowy diagnostycznej (wag) stanowi podstawę do przeprowadzenia badań ich wartości. Można przy tym zastosować jedną z następujących strategii:
- badać kolejno substraty według nadanych rang diagnostycznych,
- badać wybrane substraty grupami, ze względu na rodzaj metod ich badania, co w znacznym stopniu wiąże się z charakterem ilościowym, jakościowym lub kosztowym substratów,
- badać system informacyjny w układzie tematycznym, proponowanym przez układ faset, uwzględniając w ich ramach wybrane substraty, a w szczególnych przypadkach – całą ich zawartość.

Wybór i hierarchia przyjętych strategii badań zależy głównie od składu i możliwości zespołu diagnozującego. Nie wyklucza się również strategii mieszanej. Właściwe i rzetelne rozwiązanie tego problemu stanowi podstawę dobrej diagnozy i zabezpiecza przed skutkami błędów diagnozowania – zwłaszcza błędów praktycznych [3].

Przesłanki określania stanu i probierza stanu systemu informacyjnego

Wybór aktywnych substratów diagnostycznych, nadanie im właściwych rang oraz ustalenie metod badania ich wartości tworzą, w procesie diagnozowania, podstawę określania stanu systemu.

Traktując system informacyjny SI jako trójkę:

$$SI = \langle S, R, P_R \rangle$$

gdzie:
$S = \{S_1 ..., S_n\}$
$R \subset S \times S$
$P_R: R \rightarrow [0,1]$
przy czym:
S – to pełny zbiór stanów,
R – relacja określona na S,
P_R – miara prawdopodobieństwa określona na R,
stan systemu informacyjnego będzie opisany za pomocą zbioru wartości substratów diagnostycznych (d_i), takich, że $d_i \in D_i$,
gdzie:
D_i – oznacza zbiory substratów faset diagnostycznych:
$i = 1, 2, ..., n$.

Iloczyn kartezjański zbiorów substratów (I^D) ma następującą postać:

$$I^D = S^D \times T^D \times C^D \times O^D \times Z^D$$

gdzie:
S, T, C, O, Z – fasety substratów diagnostycznych.

Iloczyn ten wyznacza przestrzeń stanów systemu.

Klasyfikacja stanów, a zwłaszcza jej szczegółowość i głębokość powinna być w każdym przypadku dostosowana do specyfiki systemu i do odbiorcy wyników badań diagnostycznych, a ściślej – do celu diagnozowania.

Rozpatrując problem stanu systemu nie można pominąć zagadnienia stanu idealnego, czyli sytuacji, gdy substraty diagnostyczne przyjmują wartości idealne (wzorcowe).

Próby sformułowania warunków idealnego systemu informacyjnego przybierały różne formy, jednak problem ten w aspekcie procesów diagnostycznych pozostaje otwarty. W świetle przesłanek różnorodnych celów diagnozowania i związaną z nimi koniecznością stosowania wielostronnych kryteriów oceny idealnego systemu informacyjnego można stwierdzić, że nie jest możliwe ustalenie jednego, uniwersalnego wzorca systemu informacyjnego. Te substraty bowiem, które przybierają niedopuszczalne wartości w jednym systemie, mogą być przyjęte jako wystarczające i zadowalające w innym systemie.

Można zaproponować natomiast podejście do diagnostyki systemu, umożliwiające wybór wariantu probierza systemu zgodnie z celem i przyjętym kryterium diagnozowania. W tym ujęciu probierz systemu informacyjnego wyznacza stan idealny badanego systemu w wartościach substratów diagnostycznych.

Przesłanki budowy i wykorzystania probierza systemu informacyjnego można ująć następująco:
- System informacyjny znajduje się w stanie idealnym, gdy substraty diagnostyczne, wybrane do jego opisu, przyjmą wartości zadowalające. Niektóre wartości substratów mogą być odnoszone przy ustalaniu wartości zadowalających do istniejących norm. W sytuacji ograniczenia wyboru i badań systemu informacyjnego przy wykorzystaniu substratów ilościowych można zastosować metodę polegającą na porównywaniu ilościowych parametrów badanego systemu do parametrów systemu uznanego za lepszy;
- Określenie probierza systemu, czyli przyjętego stanu idealnego, odbywa się dla każdego diagnozowanego systemu indywidualnie. Ustalanie wartości idealnych (zadowalających w danym systemie) dotyczy tych substratów diagnostycznych, które zostały oznaczone jako aktywne. Obszar danego probierza wyznaczają więc te same kryteria, które były uwzględnione przy wyborze i wartościowaniu substratów, a więc głównie typ systemu informacyjnego, przyjęte kryterium oceny systemu, cel badań diagnostycznych oraz przyjęte preferencje danych typów substratów;
- W przypadku, gdy choć jedna wartość zbadanych substratów różni się od probierza bądź budzi kontrowersję w zespole diagnozującym system informacyjny, należy kontynuować działanie diagnostyczne;
- W powiązaniu z przyjętym układem faset diagnostycznych, zakłada się możliwość istnienia odpowiadających im tzw. probierzy elementarnych, przypisanych tylko wybranej fasecie tematycznej.

Obraz diagnostyczny idealnego stanu systemu informacyjnego w określonym czasie i sytuacjach przedstawić można

w następującym równaniu stanu, wiążącym między sobą fasety substratów diagnostycznych w probierz systemu:

$$S^N \vee T^N \vee C^N \vee O^N \vee Z^N = 0$$

gdzie:
N – wskaźnik podzbioru substratów diagnostycznych o wartościach niezadowalających (symptomów negatywnych),
S, T, C, O, Z – fasety substratów diagnostycznych, przy czym:

$$S^N \subseteq S,\ T^N \subseteq T,\ C^N \subseteq C,\ O^N \subseteq O,\ Z^N \subseteq Z.$$

Natomiast równanie stanu obrazujące probierz elementarny dowolnej fasety (F^N) można przedstawić następująco:

$$F^N = \sum_{i=1}^{l} d_i^N = 0$$

gdzie
d_i^N – substrat diagnostyczny reprezentujący symptom negatywny i $d_i^N = \begin{cases} 1, \text{ gdy występuje symptom negatywny} \\ 0, \text{ w przeciwnym przypadku} \end{cases}$
l – liczba substratów diagnostycznych.

Podobne ujęcie dotyczy interpretacji diagnozowanych procesów informacyjnych. Równanie idealnego stanu systemu w aspekcie procesów informacyjnych ma następującą postać:

$$D^N \vee M^N \vee P^N \vee G^N \vee E^N = 0$$

gdzie:
N – wskaźnik występowania w procesie informacyjnym symptomów negatywnych,
D, M, P, G, R – procesy informacyjne,
przy czym:

$$D^N \subseteq D,\ M^N \subseteq M,\ P^N \subseteq P,\ G^N \subseteq G,\ E^N \subseteq E.$$

Analogicznie probierz elementarny dowolnego procesu informacyjnego (R^N), może być określony równaniem:

$$R^N = \sum_{i=1}^{l} d_i^N = 0$$

gdzie
d_i^N – substrat diagnostyczny, reprezentujący symptom negatywny w danym procesie i:
$d_i^N = \begin{cases} 1, \text{ gdy występuje symptom negatywny} \\ 0, \text{ w przeciwnym przypadku} \end{cases}$
l – liczba substratów diagnostycznych.

Wykorzystanie podanych wyżej zasad pozwala na określenie tzw. pierwszego przybliżenia diagnozy klasyfikacyjnej i stwierdzenie, czy system znajduje się w stanie idealnym czy nie. Daje ono podstawę albo do zaprzestania wszelkich działań diagnostycznych w przypadku odpowiedzi pozytywnej, albo do podjęcia dalszych kroków mających na celu ustalenie i nazwanie określonego stanu, czyli określenie punktów, w których stan systemu różni się od probierza.

Stan systemu informacyjnego przeciwny do stanu idealnego proponuje się nazwać **stanem totalnie złym**, tj. takim, gdy we wszystkich fasetach, a więc i we wszystkich realizowanych procesach, występują symptomy negatywne.

Stany pośrednie między systemem informacyjnym idealnym a totalnie złym oznaczać się będą występowaniem niedomagań w realizacji określonego procesu informacyjnego lub ich grup.

Algorytm diagnostyki systemu informacyjnego

Problem diagnozy systemu informacyjnego można, zgodnie z proponowanym podejściem, sprowadzić do ustalenia algorytmu wiązania symptomów (tj. substratów aktywnych) w syndromy tak, aby uwzględniając wagi, funkcje bliskości oraz powiązania z procesami informacyjnymi określić, które procesy przebiegają nieprawidłowo i jakie substraty (przyczyny) na to wpływają.

Podanie jednej, jednoznacznej przyczyny określonej nieprawidłowości w systemie informacyjnym jest w wielu przypadkach niemożliwe, a nawet merytorycznie nieuzasadnione. Przyjmuje się więc, że za makroprzyczynę uważać się będzie łańcuch przyczyn reprezentowany przez prawdopodobne (domniemane) substraty przyczynowe, wyodrębnione z ogólnej ich liczby.

Ustalone w fasetach substraty diagnostyczne muszą być w związku z tym analizowane jako:
- substraty przyczynowe, czyli potencjalne przyczyny osiągania wartości negatywnych przez inne substraty,
- substraty negatywne (symptomy), przyjęte za takie w wyniku obserwacji i badania systemu informacyjnego i uzupełnione o wagi, obrazujące rangę diagnostyczną danego symptomu.

Istotnym zagadnieniem przy budowie algorytmu diagnostyki jest wybór metody wiązania elementów, spełniającej postulat racjonalności i przejrzystości. Jako podstawę wiązania przyjmuje się zwykle tzw. tantiemę, czyli miarę bliskości (podobieństwa) elementów. Należy przy tym stwierdzić, że wybór funkcji podobieństwa nie ma istotnego wpływu na wynik wiązania [1].

Analiza metod wiązania, w aspekcie problemu diagnozowania systemu informacyjnego, prowadzi do wniosku, że najbliższą przyjętych założeń, będzie metoda wiązania, wyprowadzana z podejścia projektodawczego, dwuaspektowego i automatycznego.

Ujęcie projektodawcze stwarza możliwość wiązania elementów z pominięciem ograniczeń wynikających ze stanu faktycznego badanego systemu. Dwuaspektowość pozwala uwzględnić dwa zasadnicze ze względów diagnostycznych kryteria wiązania elementów, tj. bliskości i ważności (wag). Proces wiązania elementów powinien być zautomatyzowany, tzn. wykonywany przez komputer.

Uwzględniając powyższe przesłanki, można przejść do omówienia algorytmu diagnostyki systemu informacyjnego. Jego schemat obrazuje rys. 1.

W opisie algorytmu zostaną wykorzystane następujące symbole i definicje:
PSN – przestrzeń symptomów negatywnych typu „substrat przyczynowy – symptom negatywny",
PBS – przestrzeń bliskości symptomów negatywnych typu „symptom–symptom",
PSY – przestrzeń syndromów typu „substrat przyczynowy – syndrom",
PPO – przestrzeń procesów typu „substrat przyczynowy – proces",
PBP – przestrzeń wiązek typu „substrat przyczynowy – wiązka",
PPD – przestrzeń przygotowawcza diagnozy,
PBD – przestrzeń bliskości syndromów i procesów,
PDC – przestrzeń diagnozy cząstkowej systemu,

PDS – przestrzeń diagnozy systemu,
D – zbiór substratów diagnostycznych
$D = \{d_1, d_2, ..., d_i\}$,
N – zbiór symptomów negatywnych
$N = \{n_1, n_2, ..., n_j\}$,
P – zbiór procesów realizowanych w systemie informacyjnym
$P = \{p_1, p_2, ..., p_s\}$,
SN – szukane syndromy: SN_1, SN_2, ..., SN_r:
są to podzbiory zbioru N spełniające warunek:

$$N = \bigcup_{i=1}^{r} SN_i,$$

WP – szukane wiązki procesów: WP_1, WP_2, ..., WP_s:
są to podzbiory zbioru P spełniające warunek:

$$P = \bigcup_{i=1}^{r} WP_i,$$

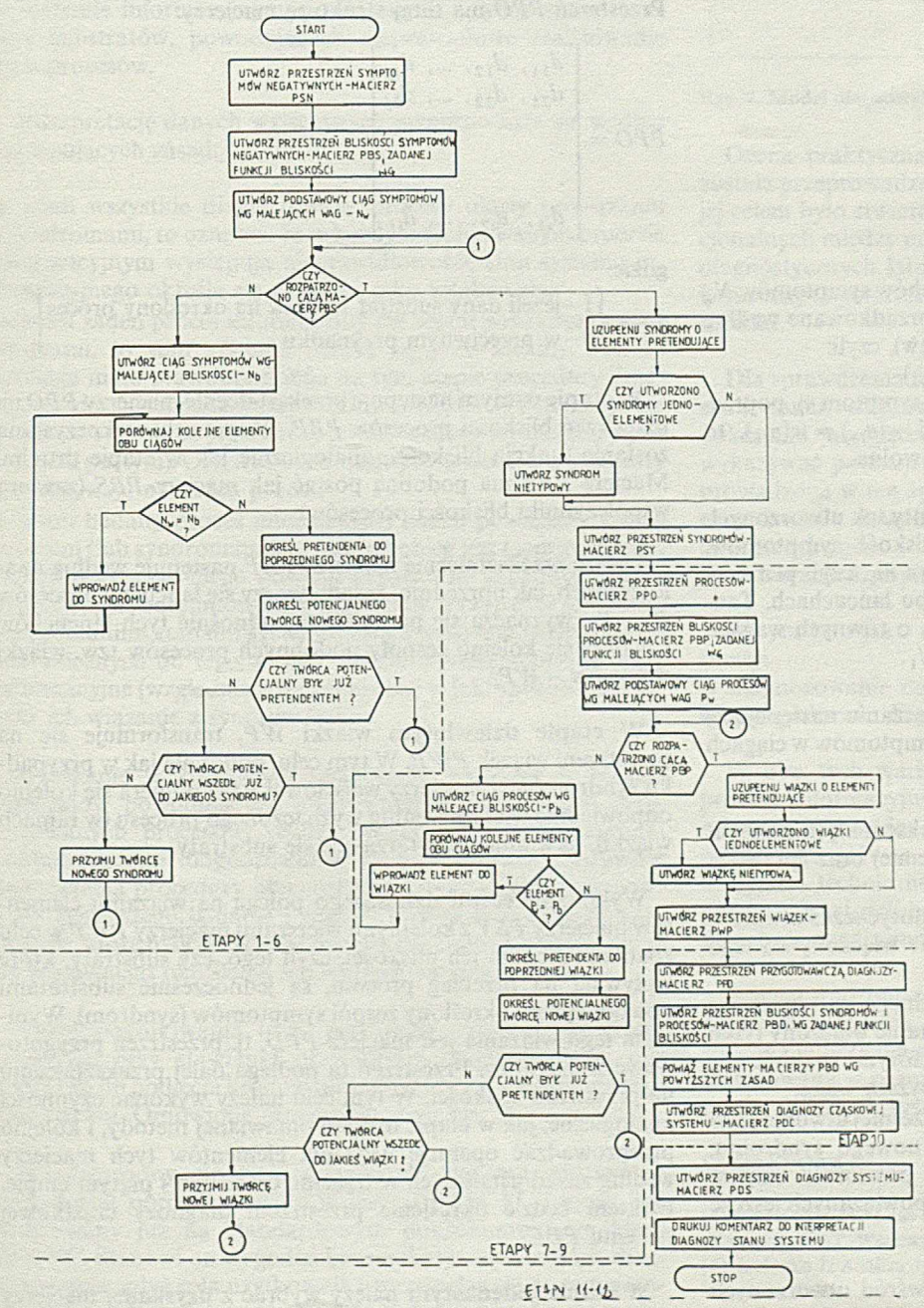
SD – szukane grupy substratów: SD_1, SD_2, ..., SD_r:
są to podzbiory zbioru D spełniające warunek:

$$D = \bigcup_{i=1}^{r} SD_i,$$

NP – szukane procesy nieprawidłowo realizowane w systemie informacyjnym: NP_1, NP_2, ..., NP_s: są to podzbiory zbioru P spełniające warunek:

$$P \leqslant \bigcup_{i=1}^{s} NP_s,$$

w – funkcja określająca wagę symptomu negatywnego,
b – funkcja bliskości symptomów negatywnych,
b_{ij} – współczynnik bliskości dwóch symptomów negatywnych,
$m_k = \min.\ i$ – parametr: minimalny wskaźnik wybierający pierwsze minimalne różniące się ogniwo łańcuchów N_n i N_b,
N_n – łańcuch symptomów negatywnych, uporządkowany według malejących wartości funkcji w (malejących wag),
N_b – łańcuch symptomów negatywnych, uporządkowany według malejących wartości funkcji b (malejących podobieństw),
P_w – łańcuch procesów, uporządkowany według malejących wartości funkcji w (malejących wag),
P_b – łańcuch procesów uporządkowany według malejących wartości funkcji b (malejących podobieństw).



Rys. 1. Algorytm diagnostyki systemu informacyjnego

Przyjęto następujące definicje:

- środek syndromu SN_r, to symptom o najwyższym stopniu ważności (wadze) w danym syndromie,
- pretendent do syndromu SN_r, to symptom ze zbioru $N - \bigcup_{i=1}^{r-1} SN_i$ najbliższy w sensie funkcji b do jej środka,
- twórca potencjalny syndromu SN_r, to symptom ze zbioru $N - \bigcup_{i=1}^{r-1} SN_i$ o najwyższej wadze,
- twórca syndromu SN_r, to symptom, który nie jest pretendem do żadnego z syndromów SN_1, SN_2, ..., SN_{r-1}.

Zasady budowy i wykorzystania algorytmu diagnostyki systemu informacyjnego można usystematyzować i ująć według następujących etapów:

Etap pierwszy polega na określeniu przestrzeni PSN na podstawie danych wejściowych, tj. wybranych substratów symptomów. Będą tutaj wykorzystane dane zawarte w fasetach tj. nazwy substratów, ich typy oraz powiązania wewnętrzne i zewnętrzne, a także dane otrzymane w wyniku obserwacji i badań systemu informacyjnego. Uwzględnione na tym etapie będą też wagi symptomów, czyli wartości funkcji w.

Natomiast symptomy są ułożone w porządku według malejących wartości wag.

Przestrzeń PSN, ujęta w formie macierzy, otrzyma następującą strukturę:

$$PSN = \begin{bmatrix} d_{11}, & d_{12}, & ..., & d_{1j} \\ d_{21}, & d_{22}, & ..., & d_{2j} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ d_{i1}, & d_{i2}, & ..., & d_{ij} \end{bmatrix}$$

gdzie:

$$d_{ij} = \begin{cases} 1 = \text{jeżeli symptom zależy od danego substratu,} \\ 0 - \text{w przeciwnym przypadku} \end{cases}$$

W drugim etapie następuje przekształcenie tej macierzy na przestrzeń bliskości symptomów, ujętą w formie macierzy *PBS*. Za podstawę przekształcenia przyjęto tu następującą funkcję bliskości:

$$b(n_i, n_j) = \frac{d_i \cap d_j}{d_i \cup d_j}$$

przy czym: $b \in \langle 0,1 \rangle$.

W aspekcie tej funkcji porównuje się pary symptomów z przestrzeni *PSN* i określa iloczyn oraz sumę logiczną elementów o wartości = 1. Wiersze i kolumny macierzy *PBS* odpowiadają symptomom, a jej elementami są wartości funkcji bliskości, tzw. współczynniki bliskości:

$$PBS = \begin{bmatrix} b_{11}, & b_{12}, & ..., & b_{1j} \\ b_{21}, & b_{22}, & ..., & b_{2j} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ b_{i1}, & b_{i2}, & ..., & b_{ij} \end{bmatrix}$$

przy czym, dla $i = j$ przyjmuje się $b_{ij} = 1$.

Etap trzeci polega na utworzeniu łańcuchów symptomów N_w i N_b. Łańcuch N_b zawiera symptomy uporządkowane według wartości funkcji b (malejących podobieństw), czyli:
n_i poprzedza n_j, jeśli $b(n_{1b}, n_{ib}) > b(n_{1b}, n_{jb})$.
Przy czym, jeżeli $b(n_{1b}, n_{ib}) = b(n_{1b}, n_{jb})$, to symptom n_{ib} poprzedza n_{jb} wtedy, gdy $w(n_{ib}) > w(n_{jb})$. Jeśli $w(n_{ib}) = w(n_{jb})$, to kolejność symptomów w łańcuchu jest dowolna.

W etapie czwartym, porównując elementy tak utworzonych łańcuchów N_w i N_b, tzn. wagi oraz bliskość symptomów, wyznacza się minimalną wartość parametru m_k, czyli pierwszy symptom mający inny wskaźnik i-ty w obu łańcuchach. Tandemy symptomów uprzednio równych (tj. o równych wagach i bliskości) utworzą wstępnie syndrom SN_1.

Etap piąty polega na cyklicznym powtarzaniu następującej sekwencji operacji aż do wyczerpania się symptomów w ciągach N_w i N_b.

Z ciągu N_w wybiera się symptomy z indeksem m_k (tj. element w wskaźniku niższym w nierównym tandemie) oraz sprawdza się, czy:

- symptom pretenduje tylko do jednego z dotychczas utworzonych syndromów SN_i; jeśli tak, to zostanie włączony do tego syndromu;
- symptom pretenduje do więcej niż jednego z dotychczas utworzonych syndromów; jeżeli tak, to zostanie włączony tylko do jednego, tj. do tego syndromu, w którym znajduje się najbliższy w sensie funkcji b symptom z wyższą wagą;
- symptom nie pretenduje do żadnego z wcześniej utworzonych syndromów; wtedy staje się on twórcą nowego syndromu; operacja ta przebiega przez wykonanie powtórnie etapów trzeciego i czwartego w odniesieniu do nie powiązanych jeszcze symptomów.

Jeżeli po tych operacjach okaże się, że wśród utworzonych syndromów są syndromy jednoelementowe (tj. zawierające po jednym symptomie), to należy je połączyć w jeden syndrom, tzw. nietypowy.

W etapie szóstym prezentowanej metody, utworzone i uzupełnione syndromy transformuje się na przestrzeń syndromów – *PSY*. Transformacja ta przebiega według następującej operacji: z macierzy wejściowej *PSN* pobiera się kolejno odpowiednie wiersze według kolejnych symptomów w ramach syndromów, likwidując powtarzające się substraty.

Siódmym etapem jest określenie przestrzeni funkcjonowania systemu informacyjnego w formie macierzy *PPO*, obrazującej zależność procesów realizowanych w tym systemie od określonych substratów. Wejście stanowią dane zawarte w festach diagnostycznych, określające wpływ substratów na realizację procesów.

Ustalenie wag poszczególnych procesów przebiega według stopnia rozwoju nieprawidłowości w ich realizacji i na podstawie przewidywanego dalszego ich biegu. Jest to tzw. diagnoza fazy, oparta na wynikach badań systemu informacyjnego. Przestrzeń *PPO* ma tutaj strukturę macierzy:

$$PPO = \begin{bmatrix} d_{11}, & d_{12}, & ..., & d_{1j} \\ d_{21}, & d_{22}, & ..., & d_{2j} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ d_{i1}, & d_{i2}, & ..., & d_{ij} \end{bmatrix}$$

gdzie:

$$d_{ij} = \begin{cases} 1 - \text{jeżeli dany substrat wpływa na określony proces} \\ 0 - \text{w przeciwnym przypadku} \end{cases}$$

W etapie ósmym następuje przekształcenie macierzy *PPO* na przestrzeń bliskości procesów *PBP*. W tym celu wykorzystana zostanie funkcja bliskości, analogicznie jak w etapie drugim. Macierz *PBP* ma podobną postać jak macierz *PBS* (zawiera współczynniki bliskości procesów).

Dalsze przekształcenie macierzy *PBP* następuje według analogicznych, jak uprzednio, zasad: tworzy się łańcuchy procesów P_w i P_b, wyznacza się parametr m_k odnośnie tych łańcuchów i tworzy się kolejno zespoły podobnych procesów tzw. wiązki procesów – WP_i.

W etapie dziewiątym wiązki WP_i transformuje się na przestrzeni wiązek *PWP*. W tym celu, podobnie jak w przypadku syndromów, z macierzy wejściowej *PPO* pobiera się kolejno odpowiednie wiersze według wyznaczonego procesu (w ramach wiązki), likwidując powtarzające się substraty.

Wykonanie **etapu dziesiątego** polega na wiązaniu elementów macierzy *PSY* z kolejnymi wierszami macierzy *PWP* w celu zbadania stopnia ich bliskości, czyli tego, czy substraty, które wpływają na przebieg procesu, są jednocześnie substratami powodującymi określony zespół symptomów (syndrom). Wynikiem tego wiązania jest macierz *PPD*, tj. przestrzeń przygotowawcza diagnozy. Przestrzeń ta podlega dalej przekształceniu na przestrzeń bliskości. W tym celu należy wykonać czynności analogiczne, jak w etapie drugim omawianej metody, i kolejno przeprowadzać operacje wiązania elementów tych macierzy według zasad ustalonych w trzecim, czwartym i piątym etapie. Efektem będzie określenie przestrzeni diagnozy cząstkowej systemu *PDC*.

W etapie jedenastym należy wybrać z uzyskanej macierzy *PDC* ten wiersz (lub te wiersze), które uległy powiązaniu

z wierszem pierwszym, reprezentującym wiązkę procesów. Wiersze powiązane z wierszem pierwszym tworzą kompleksy typu syndrom – proces – NP_s.

Należy zaznaczyć, że etap dziesiąty i jedenasty wykonywany będzie tyle razy, ile wiązek procesów zostało uprzednio wyodrębnionych. Tym samym otrzyma się tyle samo przestrzeni cząstkowych diagnozy, czyli macierzy *PDC*.

Kompleksy wybrane w etapie jedenastym utworzą przestrzeń diagnozy systemu – *PDS*.

W dwunastym etapie macierz obrazująca przestrzeń *PDS* podlega transpozycji według zasad podanych w etapie szóstym. Pozwoli to w rezultacie otrzymać informację o substratach wpływających negatywnie na badane procesy informacyjne systemu.

Zestawienie końcowe, otrzymane na podstawie macierzy *PDS*, zawiera listę procesów realizowanych nieprawidłowo w systemie informacyjnym, powiązanych z syndromami oraz listę substratów, powodujących nieprawidłowe realizowanie tych procesów.

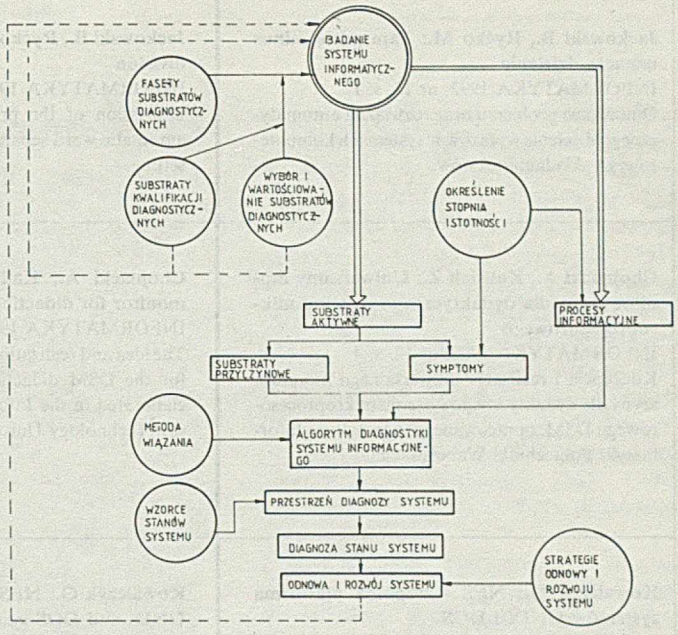
Interpretację danych wyjściowych przeprowadza się według następujących zasad:

- jeżeli wszystkie diagnozowane procesy uległy powiązaniu z syndromami, to oznacza, że w każdym realizowanym procesie informacyjnym występują nieprawidłowości; stan systemu informacyjnego określa się wówczas jako totalnie zły;
- jeżeli żaden proces informacyjny nie został powiązany z syndromami, to stan systemu ocenia się jako idealny. Jest to sytuacja mało prawdopodobna na tym etapie procedury diagnostycznej. Nie można wtedy zbudować macierzy *PDS*, gdyż nie można wyodrębnić jej wierszy (zbiór wierszy tej macierzy jest zbiorem pustym). Oznacza to, że negatywnie ocenione substraty nie wpływają na żaden proces;
- jeżeli badany proces informacyjny został powiązany z syndromem (lub syndromami), to jego przebieg jest nieprawidłowy. Oddziaływują na niego bowiem negatywnie substraty będące przyczyną wyodrębnionych symptomów. Przypadki te nazwano pośrednimi stanami systemu. Mogą mieć one wiele wariantów zależnych od tego, w jaki sposób wyodrębniono procesy informacyjne (względnie zdarzenia) oraz w jaki sposób przebiegało ich wiązanie z syndromami.

Uzyskanie w końcowym zestawieniu wyników w postaci substratów powodujących nieprawidłowe realizowanie diagnozowanych procesów, tworzy jednocześnie nowe warunki dokładniejszego diagnozowania. Istnieje bowiem możliwość powtórzenia procedury diagnostycznej, począwszy od ponownego, ściślejszego badania i zmiany wartościowania tych substratów, które były przyczyną największej liczby syndromów, aż do ponownego określenia ich wpływu na stan systemu.

Takie poszukiwania (tzw. penetracja systemu) są często niezbędne przy stawianiu prawidłowej diagnozy. Dają one podstawę do podjęcia kroków odnowy systemu, warunkującego jego rozwój. Omówione procesy ilustruje rys. 2.

Przedstawiony algorytm diagnostyki systemu informacyjnego został oprogramowany w języku PASCAL i może być realizowany już na standardowym mikrokomputerze klasy IBM PC. Stanowi on wygodne komunikatywne narzędzie diagnozowania, gdyż rola użytkownika sprowadza się do przygotowania danych wejściowych oraz do zinterpretowania zestawienia końcowego, określającego stan systemu.



Rys. 2. Model diagnostyki systemu informacyjnego

Ocena praktyczna przedstawionej metody diagnozowania została przeprowadzona na danych modelowych. Zasadniczym jej celem było stwierdzenie występowania izomorfizmów funkcjonalnych między modelem diagnozowania, a praktyką badań diagnostycznych. Istnienie tych relacji stanowi bowiem warunek konieczny i wystarczający do merytorycznej oceny poprawności modelu.

Dla sprawdzenia funkcjonowania modelu w możliwie pełnej skali badań diagnostycznych, doboru danych modelowych dokonano uwzględniając te obszary systemu, które mogą wykazywać potencjalne niedomagania. Badania zostały przeprowadzone w ten sposób, aby umożliwić:
- testowanie dowolnego obszaru (wycinka) danego systemu informacyjnego,
- testowanie kompleksowe całego systemu.

W tym celu jako egzemplifikację modelu przyjęto dwa warianty:
- diagnozowanie zasobów informacyjnych systemu,
- kompleksowe diagnozowanie systemu informatycznego.

W obu tych wariantach diagnozowaniu poddano główne procesy informacyjne realizowane w systemie. Uwzględniono substraty sytuacyjne, osiągnięć, działalności i struktury, przyjmując jako główne ujęcie diagnozowania – kryterium informacyjne i techniczno-eksploatacyjne. Jako wagi symptomów przyjęto wartości liczbowe na skali rangowej od 1 do 5. Podobnie oznaczono wagi procesów informacyjnych.

Przeprowadzone eksperymenty diagnozowania dały podstawę do stwierdzenia, że prezentowany model diagnostyki ma wymaganą przy diagnozowaniu cechę wdrażalności i może być zastosowany w różnych przekrojach i zakresach do diagnozowania złożonych systemów tej klasy.

LITERATURA

[1] Dąbrowski H., Laus-Mączyńska K.: Metody wyszukiwania i klasyfikacji informacji. WNT, Warszawa 1978
[2] Kolanko J.: Analiza dyskryminacyjna i jej zastosowanie w ekonomii. PWN, Warszawa 1980
[3] Zahedl F.: Reliability of Information System Based on the Critical Success Factors – Formulation. MIS/Qarterly, June 1987.

Jackowski B., Ryćko M.: Tam gdzie minus oznacza dzielenie
INFORMATYKA 1992, nr 12, s. 1
Omówienie problemu oraz rozwiązań automatycznego dzielenie wyrazów w systemach komputerowego składania tekstów.

Jackowski B., Ryćko M.: Where minus states division
INFORMATYKA 1992, No. 12, p. 1
Discussion of the problem and solutions for automatic word separation in computerized text setting.

Jackowski B., Ryćko M.: Dort, wo Minuszeichen Division bedeutet
INFORMATYKA 1992, Nr. 12, S. 1
Besprechung des Problems und der Lösungen von automatisierten Worttrennung in computerisierten Textsatzsystemen.

Chojnacki A., Kalisiak Z.: Uniwersalny monitor szyny dla dydaktycznego systemu mikroprocesorowego
INFORMATYKA 1992, nr 12, s. 4
Koncepcja i realizacja uniwersalnego monitora szyny dla dydaktycznego systemu mikroprocesorowego DSM, opracowanego w Instytucie Informatyki Politechniki Warszawskiej.

Chojnacki A., Kalisiak Z.: Universal bus monitor for didactic microprocessor system
INFORMATYKA 1992, No. 12, p. 4
The idea and realization of universal bus monitor for the DSM didactic microprocessor system, elaborated in the Informatics Institute of Warsaw Technology University.

Chojnacki A., Kalisiak Z.: Universelles Busmonitor für didaktisches Mikroprozessorsystem
INFORMATYKA 1992, Nr. 12, S. 4
Konzeption und Realisierung eines universellen Busmonitors für das DSM-Didaktikmikroprozessorsystem, das im Institut für Informatik der Warschauen Technischen Universität erarbeitet wurde.

Kowalczyk G.: NFS – wspólna platforma systemów UNIX i DOS
INFORMATYKA 1992, nr 12, s. 10
Charakterystyka sieciowego systemu plików NFS, umożliwiającego współpracę w sieci lokalnej komputerów o różnej architekturze i różnych systemach operacyjnych.

Kowalczyk G.: NFS – common platform of UNIX and DOS systems
INFORMATYKA 1992, No. 12, p. 10
Characteristics of the NFS network file system, which enables in local area networks cooperation of computers with different architecture and different operating systems.

Kowalczyk G.: NFS – eine gemeinsame Plattforme der UNIX-und DOS-Systeme
INFORMATYKA 1992, Nr. 12, S. 10
Eine Charakteristik des NFS-Netzdateisystems, das in Lokalnetzen eine Zusammenarbeit von Computern mit verschiedenen Architekturen und verschiedenen Betriebssystemen ermöglicht.

Szczypiński M.: Współpraca pakietu PC NFS z siecią Novell
INFORMATYKA 1992, nr 12, s. 13
Rozwiązanie problemu jednoczesnej pracy systemów NetWare i UNIX z użyciem pakietu PC NFS, bez konieczności przeładowywania systemu operacyjnego.

Szczypiński M.: Cooperation of the PC NFS package with Novell network
INFORMATYKA 1992, No. 12, p. 13
Solution of the problem of simultaneous work with NetWare and UNIX systems using the PC NFS package without necessity of computer operating system overloading.

Szczypiński N.: Zusammenarbeit des PC NFS-Pakets mit dem Novell-Netz
INFORMATYKA 1992, Nr. 12, S. 13
Eine Lösung des Problems von gleizeitiger Arbeit der NetWare- und UNIX-Systeme mit Anwendung des PC NFS-Pakets, ohne Notwendigkeit der Übersteuerung des Betriebssystems.

Surma J.: Modelowanie wiedzy w metodyce KADS
INFORMATYKA 1992, nr 12, s. 15
Charakterystyka współczesnych oraz perspektywicznych rozwiązań metodyki projektowania i realizacji systemów z bazą wiedzy.

Surma J.: Science modelling in the KADS methodology
INFORMATYKA 1992, No.. 12, p. 15
Characteristics of contemporary and future solutions for designing and realization of knowledge-based-systems.

Surma J.: Wissenmodellierung in der KADS-Methodik
INFORMATYKA 1992, Nr. 12, S. 15
Eine Charakteristik der zeitgenössischen und zukünftigen Lösungen von Projektierungs- und Realisierungsmethodik der Systemen mit Wissenbank.

Wrzeszcz Z.: Operacje wyszukiwania w bazie danych GFG
INFORMATYKA 1992, nr 12, s. 18
Charakterystyka sposobu realizacji funkcji wyszukiwania w bazie danych systemu gromadzenia informacji faktograficznej.

Wrzeszcz Z.: Searching operations in the GFG data base
INFORMATYKA 1992, No. 12, p. 18
Characteristics of the realization method for searching function in data base of factographic information acquisition system.

Wrzeszcz Z.: Aussuchungsoperationen in der GFG-Datenbank
INFORMATYKA 1992, Nr. 12, S. 18
Eine Charakteristik von Realisierungsmethode der Aussuchungsfunktion in der Datenbank des Systems zum Versammeln der faktografischen Information.

Szewczyk A.: Diagnostyka systemów informacyjnych (2). Model diagnostyki systemu informacyjnego
INFORMATYKA 1992, nr 12, s. 22
Druga część charakterystyki metod diagnozowania systemów informacyjnych, zawierająca omówienie sposobu wyboru, wartościowania oraz badania substratów diagnostycznych.

Szewczyk A.: Information system diagnostics (2). The model of information system diagnostics
INFORMATYKA 1992, No. 12, p. 22
Second part of characteristics of information system diagnosing methods, which includes discussion of diagnostic substrates selection, valuation and examination methods.

Szewczyk A.: Diagnostik von Informationssystemen (2). Das Modell von Diagnostik eines Informationssystems
INFORMATYKA 1992, Nr. 12, S. 22
Zweiter Teil einer Charakteristik von Methoden zur Diagnose der Informationssysteme, der eine Besprechung von Auswahls-, Bewertuns- und Untersuchungsmethoden der diagnostischen Substrate umfasst.

Sprzedaż promocyjna najnowszego mikrosterownika Motoroli

Tylko do 18 grudnia br. odbywa się promocyjna sprzedaż najnowszego 16-bitowego mikrosterownika 68HC16 Motoroli. Sterownik ten jest zgodny (na poziomie programu źródłowego) z bardzo rozpowszechnioną rodziną 8-bitowych mikrosterowników HC11 tej samej firmy, ale znacznie szybszy (mnożenie 16 × 16 – 480 ns, dzielenie 32/16 – 1,5 ns, krótkie rozkazy – 120 ns). Jednostkę centralną sterownika HC16 rozbudowano o funkcje cyfrowego przetwarzania sygnałów DSP (*Digital Signal Processing*), a standardowe wyposażenie stanowi m.in. 10-bitowy przetwornik analogowo-cyfrowy (A/C).

W ramach akcji promocyjnej Motorola sprzedaje zestaw obejmujący system uruchomieniowy (dołączana do PC karta, oprogramowanie, dokumentacja). Zestaw ten jest dodatkowo wyposażony w kompletny projekt przykładowej aplikaji w postaci karty FADB (*Frequency Analyser Display Board*), która po przyłączeniu do HC16 i uruchomieniu załączonego oprogramowania, realizuje funkcje analizatora widma sygnału akustycznego dla zestawu muzycznego hi-fi. Karta zawiera pięć wskaźników – każdy zbudowany z ośmiu diod LED. Sygnał akustyczny jest przetwarzany do postaci cyfrowej przez przetwornik A/C, a następnie filtrowany przez pięć zaimplementowanych programowo filtrów pasmowych DSP. Amplitudy poszczególnych prążków są wyświetlane na 8-poziomowych wskaźnikach LED. Cały projekt jest szczegółowo opisany w załączonym podręczniku *Frequency Analyser Project Manual,* a dodatkowa publikacja *Digital Signal Processing and the Microcontroller* wprowadza użytkownika w algorytmy DSP. Cena całego zestawu – poniżej 300$, mikrosterownika – 24 $.

Szczegółowe informacje można otrzymać w firmie **ELBATEX – tel. 625-48-77 w Warszawie.** ELBATEX jest jedynym autoryzowanym dystrybutorem firmy Motorola w zakresie podzespołów i systemów uruchomionych.

SYSTEC'92

dokończenie ze s. 14

Czas potrzebny na zestawienie i skoordynowanie całej ekspozycji przez firmę Softing wyniósł zaledwie dwa tygodnie. Korzystając z oprogramowania EasyMAP firmy PROCOS przeciętny student może w ciągu tygodnia odwzorować na ekranie komputera osobistego zachowanie się robota, spawającego w pobliżu laserem tłumiki samochodowe. Niezależnie od poruszającego się robota, na ekranie były wykreślane charakterysyki opisujące dynamikę jego ruchów. Szybkość, z jaką udało się osiągnąć takie rezultaty dowodzi, jakie korzyści daje posługiwanie się rozwiązaniami standardowymi.

Na sesjach EMUG Kongresu prezentowano osiągnięcia i zamiary na przyszłość w poszczególnych regionach świata. Oprócz przedstawicieli EMUG wystąpili reprezentanci regionalnych grup użytkowników MAP z Japonii, Korei, Stanów Zjednoczonych, Australii, Węgier. Nasze skromne osiągnięcia zreferował Przewodniczący Polskiej Grupy Użytkowników MAP/TOP (POMUG) Cezary Lichodziejewski.

Osobne sesje poświęcono dwum bardzo istotnym problemom, a mianowicie badaniom zgodności oferowanych wyrobów ze specyfikacją MAP oraz konieczności uzupełnienia specyfikacji MAP o protokoły i zestawy usług dla sieci typu fieldbus.

Badanie zgodności poszczególnych produktów MAP ze specyfikacją jest procesem bardzo kosztownym. Co więcej, dość popularna jest opinia, że korzystając ze znanych obecnie metod badawczych i tak w praktyce nie można zagwarantować niezawodnej współpracy dowolnych dwóch systemów, ponieważ badania takie mogą pokazać jedynie fakt wystąpienia w danej implementacji odstępstw od normy, a nie ich brak. W miejsce badania zgodności proponuje się więc testowanie wzajemnej kompatybilności poszczególnych rozwiązań. Takie z kolei rozwiązanie nie jest akceptowane przez wielu użytkowników, którzy twierdzą, że przy rosnącej stale liczbie rozwiązań jest to praktycznie niemożliwe do realizacji. Spór trwa, natomiast ośrodki naukowo-badawcze starają się zaoferować usługi, zarówno w zakresie badania zgodności z normą, jak i testowania kompatybilności.

Produkty MAP są obecnie dość drogie. Dla przykładu karta sterownika do komputera klasy PC (Carrierband 802.4) z oprogramowaniem MAP, ale bez MMS, produkcji firmy AEG Computrol kosztuje około 2400 $. Jeśli natomiast chcemy zintegrować środowisko Windows i MMS z wykorzystaniem DDE, co przy korrzystaniu z oprogramowania „MMS connect" firmy Cycle jest łatwe, to musimy liczyć się z kosztami rzędu 3500 $ plus oczywiście koszt sterownika. Koszty takie, do przyjęcia w przypadku zinterowanych systemów produkcji, są nie do zaakceptowania przy tworzeniu rozproszonych układów we-wy. Układy takie są niezbędne do komunikacji komputerów z procesem technologicznym i zwykle są tworzone ze znacznej liczby przetworników C/A i A/C, binarnych we-wy, czytników kodów paskowych, specjalizowanych wyświetlaczy, klawiatury itp. elementów, połączonych z komputerem nadrzędnym prostą i tanią siecią typu fieldbus. Obecnie istnieje wiele konkretnych rozwiązań, oferowanych przez różnych producentów (UNIBUS, PROFIBUS, P net itp.), brak jest natomiast decyzji odnośnie rozszerzenia specyfikacji MAP.

Walka z wysoką ceną produktów MAP toczy się jeszcze na innym polu. Na ostatnim sptkaniu WFMUG podjęto decyzję o takim rozszerzeniu specyfikacji MAP, aby oprócz obowiązującego do tej pory standardu Token Bus (802.4) w dolnych warstwach (1 i 2) modelu OSI umożliwić również opcjonalne stosowanie bardzo popularnego standardu Ethernet (802.3). Rozwiązanie takie będzie stanowić przełom w rozwoju MAP.

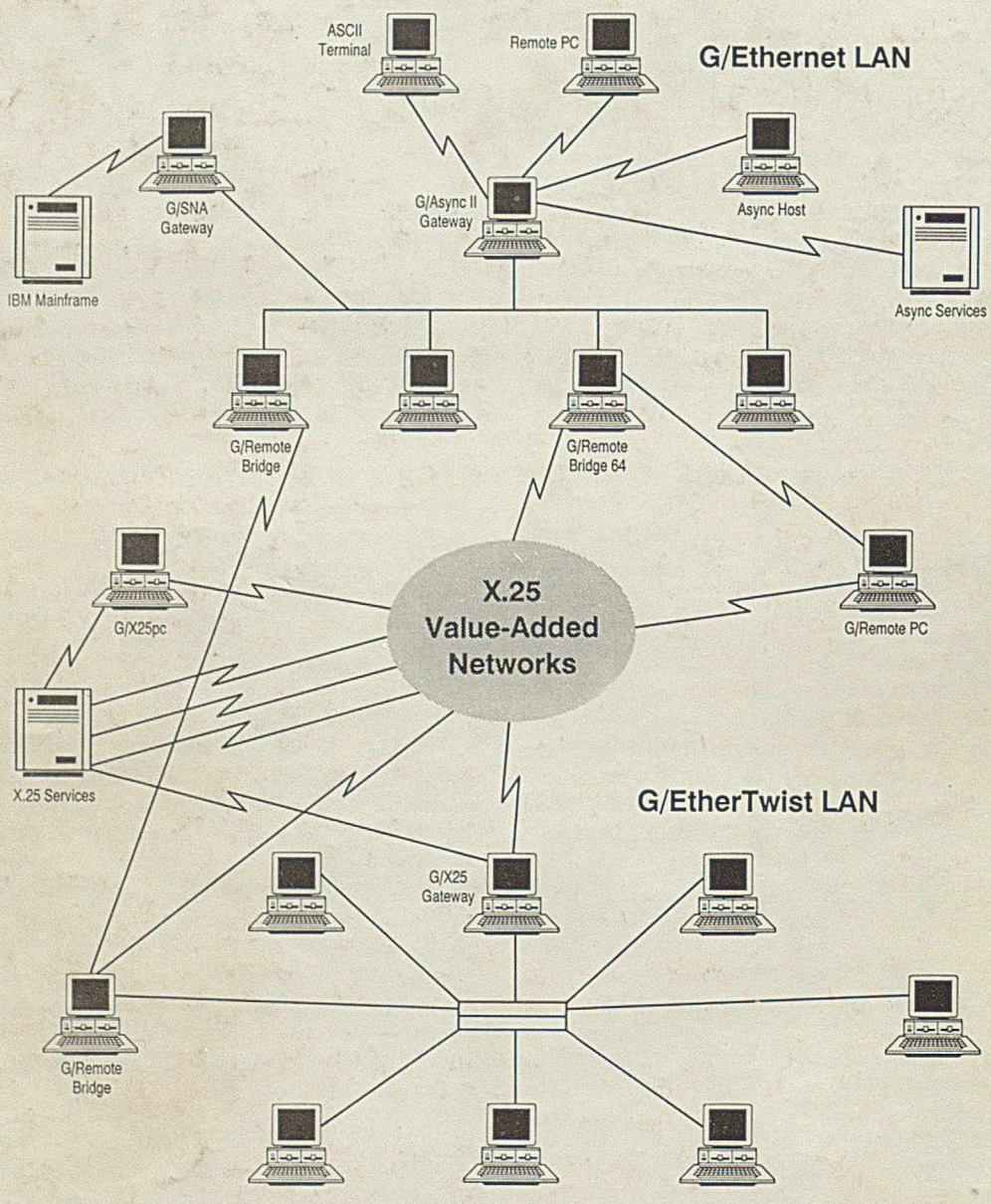
Wszystkich zainteresowanych tą tematyką proszę o kontakt telefoniczny z Autorem: tel. **Warszawa 37-50-06** lub z sekretariatem POMUG **23-82-33** lub listowny: **Instytut Elektroniki, Zakład Badań Podstawowych Elektroniki MPiH i PAN, 04-703 Warszawa, Pożaryskiego 28**

Mgr inż. MARIUSZ POSTÓŁ

BIBLIOTEKA GŁÓWNA
Politechniki Śląskiej
P.1877/92

Produkty sieciowe LA

Gateway
communications, inc.



BEZKONKURENCYJNE PRODUKTY
SIECI KOMPUTEROWYCH NAGRADZANE PRZEZ:
PC MAGAZINE, LAN MAGAZINE, INFO WORLD
OFERUJE AUTORYZOWANY DYSTRYBUTOR:

MIKROB

PC MAGAZINE EDITORS' CHOICE January 31, 1989 G/Ethernet AT
PC MAGAZINE EDITORS' CHOICE October 16, 1990 G/EtherTwist

P.W.P.T. MIKROB SP. Z O.O.
20-346 Lublin, ul. Długa 5
Tel. (0-81) 420-61, faks 415-43, teleks 643776 MIKRO PL

0/15/91

Na życzenie wysyłamy katalog produktów. Atrakcyjny program współpracy dla dealer'ów.